НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Автоматизована система менеджменту бізнес процесів на базі ITSM платформи»

Виконав:
студент IV курсу, групи ІО-64
Снітчук Максим Олегович _____

Керівник:
Доц. каф. ОТ, к. т. н.
Русанова Ольга Веніамінівна _____

Консультант з норма контролю:
Професор, доктор технічних наук
Симоненко Валерій Павлович _____

Рецензент:
Доц. каф. СП і СКС. к.т.н.
Орлова Марія Миколаївна _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Снітчуку Максиму Олеговичу

1. Тема проєкту «Автоматизована система менеджменту бізнес процесів на базі ITSM платформи», керівник проєкту Русанова Ольга Веніамінівна, старший викладач, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту *26 травня 2020р.*

3. Вихідні дані до проєкту *див. технічне завдання*

4. Зміст пояснювальної записки *Аналіз і характеристика об'єкта проектування, обґрунтування оптимального варіанта реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку. Висновки.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *приниципова схема, функціональна схема, структурна схема*

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	Сімоненко В. П., професор, д.т.н.		

7. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Затвердження теми роботи*	*01.09.2019*	
2.	*Вивчення та аналіз завдання*	*15.12.2019-15.03.2020*	
3.	*Розробка архітектури додатку*	*15.03.2020-25.03.2020*	
4.	*Написання програмної частини*	*25.03.2020-05.04.2020*	
5.	*Тестування та виправлення помилок*	*05.04.2020-15.04.2020*	
6.	*Оформлення пояснювальної записки*	*15.04.2020-20.05.2020*	
7.	*Захист програмного продукту*	*25.04.2020*	
8.	*Передзахист*	*26.05.2020*	
9.	*Захист*	*18.06.2020*	

Студент Максим СНІТЧУК

Керівник Ольга РУСАНОВА

АНОТАЦІЯ

В даній дипломній роботі було створено сервіс для автоматизації бізнес процесів для користувачів у порталі самообслуговування на базі ITSM платформи ServiceNow. У ході роботи були досліджені можливості платформи у порівнянні з її конкурентами та була проаналізована можливість оптимізації обробки користувацьких запитів в контексті платформи.

У якості серверної частини були використані API платформи. А для написання користувацьких інтерфейсів було використано бібліотеку React.

АННОТАЦИЯ

В данной дипломной работе был создан сервис для автоматизации бизнес процессов для пользователей в портале самообслуживания на базе ITSM платформы ServiceNow. В ходе работы были исследованы возможности платформы по сравнению с ее конкурентами и была проанализирована возможность оптимизации обработки пользовательских запросов в контексте платформы.

В качестве серверн части были использованы API платформы. А для написания пользовательских интерфейсов было использовано библиотеку React.

ABSTRACT

In this thesis, a service was created to automate business processes for users in the self-service portal based on the ITSM platform ServiceNow. In the process of the work, the possibilities of the platform in comparison with its competitors were investigated and the possibility of optimizing the processing of user requests in the context of the platform was analyzed.

Platform APIs were used as the server part. And the React library was used to write user interface.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6423. 02.000 ТЗ	Технічне Завдання	4	
3	А4	ДП 6423. 03.000 ПЗ	Пояснювальна записка	56	
4	А4	ДП 6423. 04.000 Д1	Принципова схема	1	
5	А4	ДП 6423. 05.000 Д2	Функціональна схема	1	
6	А4	ДП 6423. 06.000 Д3	Структурна схема	1	

					ДП.6423.01.000 ВП		
Зм.		№ документа	Підп.	Дата			
Розробив		Снітчук М.О.			Автоматизована система менеджменту бізнес процесів на базі ITSM платформи Відомість дипломного проєкту	Літ.	Аркуш
Перевір.		Русанова О.В.				Т	1
							НТУУ «КПІ імені Ігора Сікорського», ФІОТ
Н.конт		Симоненко В.П.					
Затв.							

Технічне завдання
до дипломного проєкту

на тему «Автоматизована система менеджменту бізнес процесів на базі ITSM платформи»

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Автоматизована система менеджменту бізнес процесів на базі ITSM платформи».

Область застосування: програма може використовуватися розробника на платформі ServiceNow або компаніями, що використовують дану платформу, як реалізацію ITSM

2.ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3.МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення автоматизовананої системи менеджменту бізнес процесів на базі ITSM платформи.

4.ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

5.ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Додаток, що розробляється повинен:

- Повинен бути зручним у використанні.
- Повинен мати інтерфейс з мінімальним навантаженням, для швидної орієнтації.
- Мати можливість для швидного пошуку потрібних записів.
- Мати можливість здійснювати користувацькі запити.
- Повинен працювати так само швидко або швидше аніж схожі вбудовані додатки.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення необхідної літератури	19.02.2020
Складання і узгодження технічного завдання	06.03.2020
Написання вступної частини та огляд рішень	19.03.2020
Розробка архітектури додатку	03.04.2020
Написання програмної частини	10.04.2020
Тестування та виправлення помилок	01.05.2020
Оформлення документації дипломного проекту	15.05.2020
Попередній захист та проходження нормативного контролю	29.05.2020
Захист дипломного проекту	15.06.2020

Пояснювальна записка
до дипломного проєкту

на тему: «Автоматизована система менеджменту бізнес процесів на базі ITSM платформи»

Київ – 2020 року

ЗМІСТ

ВСТУП

На даний час, через бурхливий розвиток інформаційних технологій, бізнес будь-якого рівня використовує комп'ютер та інтернет. На разі веб технології дозволяють компаніям економити, пришвидшувати роботу, отримувати кращі зворотні відгуки. Звичайний сайт-візитка допомагає клієнту знайти та обрати найкращий сервіс, хороших варіантів настільки багато у різних сферах, що можливо розгубитись у виборі серед найкращих. Малому та середньому бізнесу, обдумавши, достатньо розробити власний сервіс, як у них відразу з'являється можливість отримувати більше клієнтів та прибутку. Те, що є допомогою для малих і середніх компаній, є спасінням для компаній вище середнього рівня та компаній гігантів. Адже раніше, при роботі у великих компаніях, працівники при виникненні мінімальних проблем, могли вирішувати їх у рази повільніше ніж зараз. А все тому, що зараз для зв'язку з іншою людиною, достатньо менше однієї хвилини, що в десятки, а то і сотні разів швидше ніж раніше. Проте технології пішли ще далі, якщо раніше існували професії, де люди витрачали місяці роботи для обробки даних та для їх передачі на обробку іншим, зараз достатньо написати до двадцяти рядків коду та задати певні параметри для виконання. З поширення комп'ютерів у великому обігу, було запропоновано документ з кращими підходами до вирішення тих чи інших задач, що називається ITIL. Невдовзі теоретичний підхід був підтверджений з практичної сторони та були розроблені системи, що об'єднували в собі найкращі підходи ITIL. На сьогодні більшість компаній гігантів використовують ту чи іншу систему для автоматизації та моніторингу своїх процесів у середині компанії та за та її межами, для пришвидшення роботи між відділами та з постачальник послуг.

Таким чином новітні системи допомагають обробці різних процесів у компаніях, що значно полегшує роботу менеджменту та у певній мірі і всім іншим відділам в компанії.

РОЗДІЛ 1

ОГЛЯД ITSM СИСТЕМ

1.1 ITIL та ITSM

Бібліотека IT infrastructure library (бібліотека ITIL) або Бібліотека інфраструктури інформаційних технологій - це серія книг, що містять набір інструкцій з управління, налагодженні і постійного поліпшення бізнес-процесів, пов'язаних з IT.

Перша редакція бібліотеки на замовлення британського уряду була створена в 1986-1989 роках, і почала публікуватися в 1992 році, а остання, третя версія, - ITIL V3 - вийшла в 2007 році. Остання редакція бібліотеки, що вийшла в 2011 році, складається з 5 томів. На початку 2019 року вийшов провісник четвертої версії бібліотеки V4, повну версію якої розробник AXELOS випустить орієнтовно через рік. Після подальшої стандартизації та структуризації всіх бізнес процесів і галузях IT, був створений підхід для управління цими процесами, який називається ITSM. На базі цього підходу були створені рішення, для управління всіма бізнес процесами всередині однієї платформи.

При розробці третьої редакції був використаний новий підхід до формування її змісту, так званий «життєвий цикл послуги». Його суть полягає в тому, що кожен том бібліотеки фокусується на певній фазі «життєвого циклу». Так як фаз цього циклу за версією бібліотеки ITIL п'ять, то і книг, які в ній містяться, теж п'ять[18]:

- Стратегія послуг (Service Strategy);
- Проектування послуг (Service Design);
- Перетворення послуг (Service Transition);
- Експлуатація послуг (Service Operation);
- Постійне поліпшення послуг (Continual Service Improvement).

1.2 Інструменти ITSM платформ

Методологія ITSM робить IT-відділ сервіс-провайдером для інших підрозділів організації. Він перестає бути допоміжним елементом, відповідальним за підтримку працездатності IT-інфраструктури: окремих серверів, мереж і додатків.

Компанія формалізує послуги, які вона бажає отримувати від IT-відділу, і переходить на роботу за моделлю «замовник - постачальник». В результаті бізнес починає висувати свої вимоги до сервісів, формулюючи проблеми і завдання, з якими стикаються користувачі. А IT-відділ вже сам вирішує, за допомогою яких технічних засобів задовольнити ці потреби.

Основними інструментами ITSM вважають:

- Управління активами (ITAM, IT Asset Management). Це - процес, який відповідає за облік IT-активів на всьому протязі їх життєвого циклу: від придбання або розробки до списання. До IT-активів в цьому випадку відносяться різного роду програмне і апаратне забезпечення: ПК, ноутбуки, сервери, оргтехніка, інтернет-ресурси. Автоматизація управління активами дозволяє компанії ефективніше витрачати ресурси і прогнозувати потреби.
- Управління фінансами (ITFM, IT Financial management). Це - процес, частиною якого є оптимізація IT-послуг з економічної точки зору. IT-відділу і організації необхідно збирати фінансову інформацію для розуміння загальної картини витрат і доходів.
- Управління та моніторинг ЦОД (ITOM, IT Operations Management). Мета цього процесу - моніторинг компонентів IT-інфраструктури та балансування навантаження. Фахівці IT-відділу повинні розуміти, як зміна продуктивності сервера або мережевого комутатора відіб'ється на якості послуг, що надаються.
- Портал самообслуговування (Service Portal). Такі портали дають користувачам можливість самостійно вирішити свої проблеми з програмним або апаратним забезпеченням, не вдаючись до допомоги

фахівців техпідтримки. Є кілька варіантів побудови таких порталів - статичні бази знань, FAQ або динамічні сторінки з можливістю прийому заявок. [1]

- Управління інцидентами (ITIM, IT Incident Management) – це процес, спрямований на усунення будь-яких інцидентів, що викликають переривання IT-послуг, найшвидшим і ефективним способом. Згідно ITIL інцидентом є, як несправність в роботі програмного або апаратного забезпечення, так і будь-яке відхилення від узгоджених з користувачем параметрів надання IT-послуги, яке призводить до припинення або зниження якості IT-сервісу. Основні завдання процесу управління інцидентами:
    - Виявлення і реєстрація збоїв в наданні IT-послуг.
    - Класифікація інцидентів.
    - Призначення завдань IT-персоналу, який відповідає за відновлення цих IT-послуг.
    - Контроль відповідності часу закриття інцидентів параметрам, визначеним в угоді про рівень сервісу (SLA)[16]

1.3 Приклади реалізацій ITSM платформи та їх відмінності.

В даній роботі в якості платформи для розробки продукту ми будем використовувати ServiceNow, це платформа, яка є прикладом імплементації ITSM. Цей додаток об'єднує в собі функціонал яких вирішує потреби компаній які користуються інформаційними технологіями для автоматизації та оптимізіції виконання бізнес процесів таких як:

- Комунікація між компанією та клієнтом або комунікація всередині компанії.
- Хмарні зберігання даних та маніпуляція ними.
- Облік даними компанії, наприклад облік працівників.
- Можливість управління власними процесами

Проте на ринку зараз присутня велика кількість ITSM систем або сервісів. Та в останній час такі сервіси розробляють не тільки у США або Китаї, а і на Україні. Вони загалом різняться за кількома критеріями, а саме величина бізнесу, його напрямок, кількість співробітників в компанії. В даній роботі, розглядаючи ServiceNow, ми будемо порівнювати з її основними конкурентами в її області призначення, а саме здебільшого великий бізнес.

Нижче перераховані найпопулярніші ITSM системи:

- **Servicenow**
- **SolarWinds Service Desk**
- **Freshservice**
- **Wrike**
- **SysAid**
- **Zendesk**

Нижче представлені таблиці з порівняльними характеристиками:

- Операційні системи або платформи

Табл. 1.1 –Порівняння систем за доступними платформами.

ITMS Система	**Операційна система**
Servicenow	Будь-яка (використовується у браузері)
SolarWinds Service Desk	Windows, Mac, Linux.
Freshservice	Windows, Mac, Linux, Android, iOS.
Wrike	Windows, Mac, Linux, Android, iOS.
SysAid	Windows, Mac, Linux.
Zendesk	Windows, Mac, Linux.

- Імплементовані інструменти

Табл. 1.2 –Порівняння систем за реалізованими інструментами.

ITMS Система	Перелік інструментів
ServiceNow	Incident Management, Problem Management Performance Analytics, Request Management, Service Catalog, Knowledge Management, Asset and Cost Management, Self-Service portal
SolarWinds Service Desk	Incident Management, Service Portal, Change Management, IT Asset Management, Problem Management, Knowledgebase.
Freshservice	Incident Management, SLA Management, Knowledge Management, Service Catalog, Self-service portal, Team Huddle, & Automation.
Wrike	IT service management templates, Interactive Gantt charts, Custom workflows, etc.
SysAid	Incident Management, Problem Management, Change Management, Service Level Management, CMDB, Self-service portal, Knowledge Management.
Zendesk	Ticketing System, Knowledgebase, Help Desk Software, Security.

- Ціна та безкоштовний період випробовування

Табл. 1.3 –Порівняння систем за ціною та пробним терміном.

ITMS Система	Ціна	Випробовування період
Servicenow	Починаючи з $10000.00 за рік	Немає
SolarWinds Service Desk	Від $15 в місяць за користувача	30 днів
Freshservice	Від $19 до $99 в місяць за користувача.	21 днів
Wrike	Безкоштовно для 5 користувачів, від $24.80 в місяць за користувача.	Присутній
SysAid	Від $1,211 за 500 активів та 5 користувачів за рік та $1,611 за 1,000 активів за рік.	30 днів
Zendesk	Від $5 в місяць за користувача.	Присутній

З порівнянь наведених вище зрозуміло що платформа ServiceNow доступна з усіх платформ, так як її основний продукт – це веб сервіс, що доступний з будь-якого браузера. Проте Freshservice та Wrike доступні в більшості популярних платформ, що робить їх доступними для більшості. З можливих ITSM інструментів в ServiceNow імплементована більшість в порівнянні з іншими, що і відповідає її найвищою вартістю серед конкурентів. Попри велику кількість готових інструментів, слід зазначити, що дана система добре масштабується при побудові власних додатків та конфігурується за власними потребами, як цього вимагає бізнес. У ServiceNow немає явного обмеження у кількості користувачів, та як було зазначено вище її базові можливості можна розширити, що з однією сторони робить її дуже гнучкою, проте таких підхід може програвати у швидкості обробки запитів через велику кількість процесів, які працюють у фоновому режимі.

Загалом для середнього та в деяких випадках для великого бізнесу такі системи як Freshservice, SolarWinds Service Desk або Wrike буде чудовим вибором, так як більшість потрібних інструментів там також реалізовані і

системи теж мають змогу масштабуватись власними додатками. Явною перевагою ServiceNow, це найбільше співтовариство розробників та адміністраторів, через що платформа все ще є лідером на ринку.[4]

1.4 Задачі порталу самообслуговування

Порталами називають користувацькі сторінки, які різняться від загального виду системи, їхнє представлення створюється як окремі сторінки за допомогою HTML, CSS та JavaScript, на платформі для зручної розробки користувацьких інтерефейсів, підтримувався лише фреймворк Angular. Проте з різким розвитком платформи на разі в якості бібліотеки для розробки представлення можна використовувати також React, Vue, як було вказано вище. В ServiceNow системи самообслуговування (Self-Service), один з інструментів ITSM, реалізований через портали, тобто сторінки, що дозволяють клієнту знайти відповідь на своє питання (рішення своєї проблеми) без звернення до служби підтримки. Такі системи економлять час та гроші за рахунок зниження навантаження на відділ підтримки.[17]

Автоматизовані інструменти вирішують поширені проблеми і розбираються з низькорівневими заявками. Таким чином відбувається «природний відбір» тікетів. Служба підтримки може зосередитися на допомозі клієнтам з серйозними питаннями, які вимагають нетривіального рішення і вищого пріоритету.

Ще одна перевага - самообслуговування підвищує доступність підтримки. Три чверті всіх саппорт-відділів не працює цілодобово. В цьому випадку, якщо проблема виникла у вихідний день, вона буде лежати невирішеною до понеділка. Портали самообслуговування доступні в будь-який час доби. Крім того, користувачі звикли отримувати відповідь на своє питання за кілька хвилин і готові платити за це гроші.

Самообслуговування виключає всі фактори зволікання і максимально знижує час обробки заявки. До цього ж переважно можна додати розширення географії

підтримки. Підготовка порталу для самостійного вирішення проблем на декількох мовах замінює собою необхідність найму їх носіїв.

Системи самообслуговування в ITMS можуть бути декілька типів, вони різняться підходом до вирішення питань та рівнем автоматизації. Виділимо наступні:

- Статична база даних та FAQ - Ці інструменти грунтуються на двох припущеннях: що конкретний користувач повинен знати і що він захоче дізнатися в майбутньому. Компанія передбачає можливі сценарії на підставі раніше створених заявок і часто задаються. З точки зору ITSM, це може бути база кращих практик, яким співробітники повинні слідувати.
- Динамічний портал – Це інструмент, який працює в напівавтоматичному режимі. Він надає можливість формування заявки з інтелектуальною системою підбору відповіді з бази знань. Приклад - корпоративний портал самообслуговування на платформі ServiceNow. Головна перевага такого інструменту - централізація. Коли всі сервіси розташовуються в єдиній бібліотеці, користувачам стає набагато легше знайти відповідь на своє питання. Динамічним може бути і розділ з часто вживаними питаннями.
- Інтелектуальний помічник - 71% споживачів віддає перевагу віртуального помічника для пошуку інформації. Компанії-гіганти інвестують мільйони в чат-боти. Комп'ютерні агенти замінюють персонал на обслуговуванні простих завдань, черпаючи відомості з бази знань. Якщо бот не може знайти рішення для інциденту, він автоматично створює заявку на основі деталей з бесіди з користувачем.
- Гібридний чат – Іноді прийняти заявку клієнта слід ще до того, як чат-бот зайшов в глухий кут. У таких випадках знадобиться технологія гібридного чату. Вона дозволяє миттєво перемкнути користувача на живу людину і продовжити розмову з того місця, де закінчив бот.[8]

1.5 Особливості платформи ServiceNow.

ServiceNow – це платформа, яка дає можливість хмарного храніння й обчислення даних. В середині платформи даними є все, починаючи з чисел та рядків, закінчуючи скіптами, які виконуються за потребою. Для користування платформою можна придбати instance (надалі екземпляр ServiceNow, або просто екземпляр) в кількості від двох штук.

Екземпляр - це незалежна реалізація ServiceNow, самостійне середовище для використання клієнтом, яке має можливість разширятись власними програмними додатками, налаштовуватись за потребами. Він ізольований і автономний, тобто ваш примірник не надається спільним з іншими клієнтами, тобто ви можете зробити все що хочете з ним: змінити логіку, оновити інтерфейс користувача або додати поле в таблицю. Сутність це кілька речей, по перше це ваша окрема URL-адреса (щось на зразок https://<ім'я_сутності>.service-now.com/), також це програмне забезпечення, що працює в хмарі; це персональна копія платформа ServiceNow. ServiceNow надає платформу та набір додатків як послугу.

Буває два типи таких середовищ – це Production та Subproduction. У кожного замовника є дві або більше сутностей; знову все ізольовано і незалежно. Один

екземпляр може бути позначений для розробки на іншому, для тестування, а інший для

кінцевого використання. І оскільки кожен екземпляр є незалежним, кожен може працювати з різними версіями ServiceNow.

Для розробки використовується мова JavaScript версії ES5, для написання як серверної частини так і клієнтської.

- Для серверної частини для JavaScript, використовуються внутрішні бібліотеки написані творцями платформи, для манімуляціями з таблицями, роботою з числами і інші. Також є можливість підключати деякі зовнішні бібліотеки.

- Для клієнтської частини, точніше сказати для розробки клієнтских інтерфейсів, використовується фреймворк Angular версії 1.7.

Продукт має для використання декілька інтерфейсів, які слугують середовищем для різних типів користувачів:

1) Власне платформа, яка є первинним середовищем для всіх користувачів. Через яку здійснюється доступ до всіх таблиць з даними та до форм через які створюються самі дані.
2) Мобільний агент – мобільний додаток, що має доступ до найважливіших таблиць, дозволяє швидко відповідати на запити та комунікувати у внутрішньому чаті.
3) IU Portals - це середовища для кінцевих користувачів мають, створюються для того, щоб зкомпонувати автоматизовані рішення та вивести їх на зручному інтерфейсі.

За управлінням доступу в системі відповідає підхід *RBAC* (англ. *Role Based Access Control, RBAC*) – за яким кожен користувач має масив ролей, які визначають доступи, що має кожен користувач (в більшій мірі визначається доступ до таблиць, колонках на таблицях, формах, API та інші).

1.6 Дані та маніпуляції з ними

Головне що потрібно знати про дані в платформі – майже все в платформі зберігається в таблицях, наприклад потрапивши на головну сторінку (рис.1), яка складається з логотипу, верхньої контрольної панелі, списком додатків та основної частині в якій в даному випадку відображається графік, все окрім панелі контролю зверху справа зберігається в таблицях, я дані.

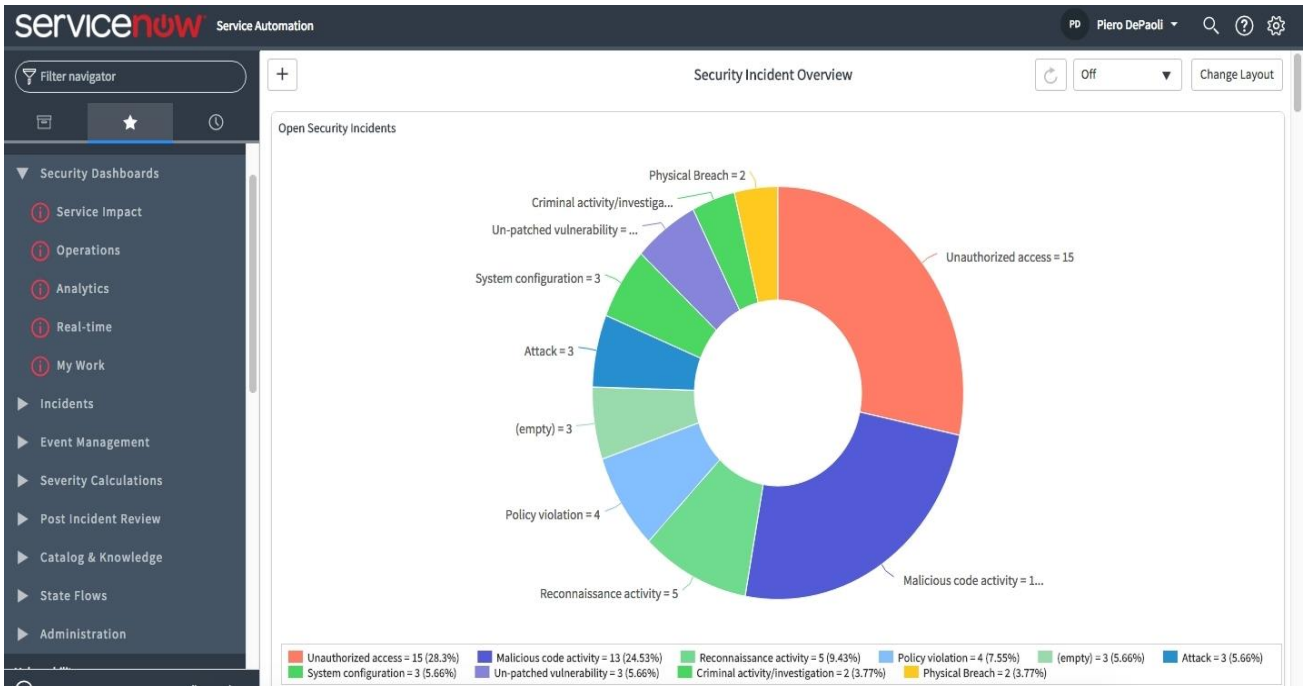


Рис. 1.1 Вигляд домашньої сторінки платформи

Через те що всі таблиці запрограмовані на верхньому рівні, це означає, що не потрібно перезавантажувати сервер, коли був імплементований новий функціонал, достатньо просто зберегти відповідний запис в таблиці.

ServiceNow може вважатися платформою високого рівня, яка базується на концепції Model-View-Controller. При створенні додатоку ServiceNow, спочатку потрібно подумати про структуру даних. Визначається, яку інформацію вам потрібно зберігати і як це все пов'язує разом, створюючи таблиці та поля, це аспект моделі. Автоматично система будує форми та списки, що показують інформацію. Далі будуються прості способи маніпуляції та зміни даних, за допомогою автоматизації та простих ручних оновлень, це аспект контролера.

Таблиці в системі створюються в таких самих формах як і інші записи, також таблиці зберігаються в окремій таблиці, яка як і деякі інші вважається системною. Системні таблиці, на відмінну від користувацьких, неможливо видалити, проте можливо змінити, точніше сказати їх можна доповнити новими полями.

При створенні користувацької таблиці автоматично створюються шість системних полів. Вони включають дві дати (коли створено запис і коли він

востаннє оновлений), два рядкові поля (містить ідентифікатор користувача того, хто його створив і який оновив), унікальний GUID sys_id та поле, що підраховує кількість оновлень запису. Вони всі оновлюється автоматично, і зазвичай нормальна практикою вважається не змінювати їх.[10]

ServiceNow підтримує реляційну модель бази даних. Це означає, що один біт даних може відноситься до іншого. Щоб забезпечити посилання на кожен запис чи рядок легко, кожен запис має унікальний ідентифікатор: первинний ключ. У ServiceNow цей первинний ключ - це те, що не пов’язано із самими даними. це є глобальний унікальний ідентифікатор або GUID. Цей GUID зберігається як 32-символьний рядок, зроблені з шістнадцятковій системі числення (цифри 0-9, плюс літери a-f). Кількість унікальних значенень GUID потенційно настільки велика, що ймовірність того, що в двух випадках буде однакове значення дуже мала. Якщо ж трапиться так, що згенерований ідентифікаційний номер вже є в системі, тоді він буде згенерований знову і так поки не знайдеться унікальне значення.

Екземпляри ServiceNow в якості бази даних використовують MySQL, це популярна база даних з відкритим кодом, яка є надійною та масштабованою. Реляційні бази даних прості в розумінні, що є однією з причин чому їх найчастіше використовують; дані зберігаються у табличному форматі з кожною таблицею зберігання інформації про певний предмет. Між ними можуть існувати відносини предметів. Відносини бувають трьох типів:

- Зв’язок "один-до-багатьох"
- Зв’язок "багато-до-багатьох"
- Зв’язок "один-до-одного"

1.7 Програмування у ServiceNow.

Мовою для написання скріптів в даній системі, для маніпуляції з даними та написання користувацьких інтерфейсів є JavaScript. Що не дивно, якщо глибше заглянути в історію створення цієї мови, адже насамперед вона

розроблялась за Event-Driven Programming парадигмою, що перекладається, як Подійно-Орієнтоване Програмування. Щоб зрозуміти цю парадигму достатньо згадати як працювала ця мова спочатку, а саме у зв'язці з мовами CSS та HTML, де браузер виконував код JavaScript, певними функціями у відповідь на взаємодію з інтерфейсом. Якщо для взаємодії з IU вище зазначена мова є в певній мірі монополістом, то для виконання серверних операцій вона скоріш за все використовуєть через зручність для бізнесу, ажде використовувати одну мову для написання усього додатку дуже вигідно для бізнесу.[5, с. 52]

Якщо в браузері JavaScript виконує двигун всередині браузера, на стороні сервера платформи ServiceNow усі скріпти виконує двигун Rhino. Сам Rhino написаний на Java, що забезпечує натяк на те, що написаний код бекенд-платформи, на який написано ServiceNow. Rhino забезпечує повну реалізацію мови, за винятком деяких об’єктів, що має сенс для клієнта. Він відносно повний і сумісний із стандартами. ServiceNow включає в себе версію Rhino, яка підтримує версію JavaScript 1.5 [5, с. 54]

Як було визначено раніше скріпти бувають серверні та клієнтські, якщо другі виконують нескладну логіку взаємодії IU з клієнтом, то основні серверні діляться залежно від того, коли вони виконуються на наступні типи:

- Business rule: серверний тип скрипта який спрацьовує від певної дії з записами певних таблиць, тобто скрипт який буде виконуватись після вставки, оновлення, зчитування або видалення, наприклад можна налаштувати таким чином, що після видалення запису з однієї таблиці буде створюватися запис в другій.
- Scheduled Job: серверний тип скріпта який можна налаштувати виконуватися залежно від дати та в заданий час, тобто можна налаштувати скріпт виконуватися за замовчуванням кожен день, один раз в місяць, один раз в два тижні і т.д.
- UI Action: це кнопка на формі або списку записів, при натисканні на яку, виконується скріпт.

- Script Include: серверний скрипт, який не виконується самостійно, але його можна викликати всередині інших скриптів, тобто основною метою є винесення логіки в класи та функції, також в дані включення можна виносити вихідний код різних бібліотек щоб використовувати в інших скриптах.
- Scripted REST API: серверний скрипт, що спрацьовує у відповідь на певний HTTP запит на адресу, що можна задати. Цей скрипт дозволяю доступатися до даних екземпляру з клієнтських додатків, або спеціальних програм для створення HTTP запитів.

Клієнтські скріпти загалом бувають двох типів, які мають дві різні цілі для використання.

- Перші виконуються на стороні системи, а саме на формах, які виконують загалом прості функції з внутрішньої бібліотеки GlideForm, з яких зокрема, заповнення полів, заборону на редагування, робити поля прихованими та робити поля обов'язковими до заповнення.
- Другі, це клієнтські скріпти, які контролюють клієнтські інтерфейси на порталах для користувачів.

1.8 Додаткові можливості

Окрім шикорого вибору ITSM інструментів, слід окремо виділити увагу, іншим інструментам та можливостям, котрі роблять систему зручною та безпечною у використанні, серед них:

- Функція єдиного входу (SSO) – система, яка дозволяє авторизуватись у платформу використовуючи різні провайдери ідентифікації, найпопулярніший приклад Microsoft. SSO дозволяє користувачеві входити в програму, не надаючи ідентифікатора користувача або пароля. Він використовує ідентифікатор Windows і пароль Windows.

- LDAP (англ. Lightweight Directory Access Protocol – це мережевий протокол прикладного рівня для надсилання запитів та модифікації даних служби каталогів через TCP/IP.
- ServiceNow забезпечує можливість оркестрування або автоматизації простих або складних завдань на віддалених серверах. Після того, як Оркестрація буде впроваджена в будь-якій IT-компанії, вся робота потребує меншої кваліфікації та праці. Він може автоматизувати такі системи, як VMware, поштові сервери Microsoft Exchange тощо.
- Платформа забезпечує можливість публікації або споживання API одночасно. SOAP, WSDL або REST API підтримуються протоколами. Ви можете створити API без коду або сценарій.
- Наявність мобільного додатку, наразі є обов'язком для успішного бізнесу, через це слід окремо акцентувати, що можливість використовуючи смартфон, вирішувати бізнес процеси віддалено від свого персонального комп'ютеру є дуже зручно, а інколи і необхідно. У ServiceNow є два мобільні додатки, перший для ITSM (вирішення інцидентів та задач), а інший більш універсальний, від дозволяє створювати та змінювати записи у базах даних.[12]

1.9 Обмеження платфоми

Окремо слід зазначити, що платформа попри свої переваги, має певні обмеження, які варто враховувати при використанні як користувач або розробки. Насамперед слід зазначити, що код Javascript, що використовується для написання скриптів у платформі, виконується за допомогою інструменту 'Rhino', який на відмінну від 'Node' підтримує Javascript лише версії ES5. Таким чином нові можливості мови будуть не доступні для розробника, що в певній мірі знижує швидкість написання коду. Вирішення цього недостатку є використання Babel, як транспілятор коду у потрібні версії.

Також через те, що платформа представляє собою сервер з таблиця, що може виконувати JavaScript код, у розробників немає можливості оптимізувати обробку даних сервера, тому використовуючи API хмарних сервісів, таких як

Amazon або Azure більшість часу роботи буде займати запис у таблиці. Тому в роботі з великим об'ємом даних слід враховувати можливості серверів ServiceNow.

Також варто зазначити, що платформа може працювати з обмеженою кількістю розширення файлів, та внутрішніх бібліотек не завжди вистачає для обробки різних даних. Таким чином створення та робота з файлами з розширенням '.xls' та похідних від нього, буде неможливим, тому слід використовувати формат '.csv'.

1.10 Постановка задачі

В даній роботі буде розроблено власний портал самообслуговування на базі платформи ServiceNow. З попереднього розділу видно, що портали самообслуговування реалізовані майже у всіх ITSM системах, проте основна задача в тому щоб розробити власну реалізацію використовуючи передові інструменти для розробки веб сторінок. Ця можливість з'явилась відносно недавно, після виступу розробників платформи на конференції «CreatorCon 2019», де був презентований концепт, за яким будь-які фреймворки можуть бути використані для розробки користувацьких інтерфейсів. Власне конференція була проведена в грудні 2019 року, таким чином можливості, які були презентовані ще досі не використовуються розробниками платформи та компаніями, які розробляють додатки на базі платформи, тому технології, які будуть використовуватись в даній роботі, вважаються новими та в певній мірі ненадійними з кількох причин:

- Через новизну технології, комуна розробників ще досить мала, таким чином частина проблем з якими зіткнеться розробник, потребуватиме його власного рішення.
- Через малу кількість розробників, рішення по розробці не матимуть найкращих практик, як вирішити найкращим чином задачі. Через те що не буде уніфікованого підходу, можлива велика кількість помилок.

Незважаючи на можливі труднощі при розробці, новітні UI бібліотеки та фреймворки пришвидшують процес розробки та дають змогу структурувати додатки таким чином щоб його було зручно розширяти, тому швидкий перехід до використання таких технологій де це можливо, очевидний.

Основною функцією порталу самообслуговування є можливість находити відповіді на свої запитання, тому він повинен мати зручну навігацію та логічний з точки зору користувача інтерфейс, тому важливою частиною проекту є проектування інтерфейсів. Другою функцією, є можливість виконувати більшість процесів платформи, використовуючи зручні інтерфейси, причому для кінцевих користувачів, портал повинен мати все необхідне, що є у платформі. Тому другою важливою задачею, є розробка програмних інтерфейсів або API, для доступу до ресурсів (таблиць) платформи.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

У ході написання першого розділу були розглянуті сучасні ITSM платформи та їх можливості. Були виведені плюси та мінуси найпопулярніших систем у ході чого було виведено, що при потребі обслуговувати компанію з великою кількістю працівників, один з найкращих варіантів є ServiceNow.

Також були проаналізовані можливості ServiceNow для розробки власних додатків та було розглянуто, як працює система, з чого вона складається тощо. Було розглянуто майже усі основні аспекти платформа, принципи роботи з даними, можливості писати та виконувати скрипти за будь-яких тригерів. Таким чином потреби будь-якого бізнесу можна задовольнити та як платформа має можливість створювати свої власні API, та доступатися до зовнішніх, що є дуже зручним за потребою використовувати будь-які мікро-сервіси.

В кінці роботи також були розглянуті обмеження платформи так її недоліки, що можуть завадити деяким аспектам розробки, у ході аналізу було вияснено, що головними перешкодами можуть стати обмеження у роботі з файлами різних розширень та через роботу з даними на високому рівні, швидкість операцій з базами даних значно знижується.

РОЗДІЛ 2.
СТВОРЕННЯ ВЛАСНОЇ СИСТЕМИ МЕНЕДЖМЕНТУ НА ПЛАТФОРМІ SERVICENOW

2.1 Огляд технологій для створення клієнтської частини

2.1.1 Вибір основної бібліотеки або фреймворку

Поставивши задачу в попередньому розділі, постає питання у виборі технологій, а саме основної бібліотеки або фреймворку для розробки фронт-енд частини. Серед популярних на цей час, виділимо три найпопулярніші:

- Angular, розроблений Google, вперше був випущений у 2010 році, що робить його найстарішим із партії. Це рамка JavaScript на основі TypeScript.
- React, розроблений Facebook, спочатку був випущений у 2013 році. Facebook широко використовує React у своїх продуктах (Facebook, Instagram та WhatsApp). Поточна стабільна версія - 16.X, випущена в листопаді 2018 року (з тих пір зменшуються додаткові оновлення).
- Vue, також відомий як Vue.js, є наймолодшим членом групи. Він був розроблений колишнім співробітником Google Evan You у 2014 році. За останні три роки Vue помітно змінився у популярності, навіть не маючи підтримки великої компанії.

У ході вибору однієї з вище вказаних бібліотек важливо зрозуміти, яку архітектуру матиме веб додаток. На даний час для побудови веб додатків використовують два різні архітектурні підходи:

- SPA (Single Page Application), або додаток з однією сторінкою.

Односторінковий додаток - це програма, яка працює всередині браузера і не потребує перезавантаження сторінки під час використання. Ви використовуєте цей тип додатків щодня. Наприклад, це Gmail, Карти Google, Facebook або GitHub.

SPA-комплекси - це обслуговування видатного UX, намагаючись наслідувати "природне" середовище у браузері - без перезавантаження сторінок, без зайвого часу очікування. Це лише одна веб-сторінка, яку ви відвідуєте, а потім завантажує весь інший вміст за допомогою JavaScript, від якого вони сильно залежать.

- MPA (Multi Page Application), або багатосторінковий додаток.

Багатосторінкові програми працюють "традиційним" способом. Кожна зміна, наприклад. відображення даних або повернення даних на запити сервера, що надають нову сторінку з сервера в браузері. Ці програми великі, більші, ніж SPA, тому що вони повинні бути. Завдяки кількості вмісту ці програми мають багато рівнів інтерфейсу користувача. На щастя, це вже не проблема. Завдяки AJAX, нам не потрібно турбуватися, що великі і складні програми повинні передавати багато даних між сервером і браузером. Це рішення покращується, і дозволяє оновити лише окремі частини програми. З іншого боку, це додає більшої складності і його складніше розробити, ніж односторінковий додаток. Також таких тип додатку підтримує шаблон проектування MVC, таким чином це значно полегшує масштабування великих додатків.

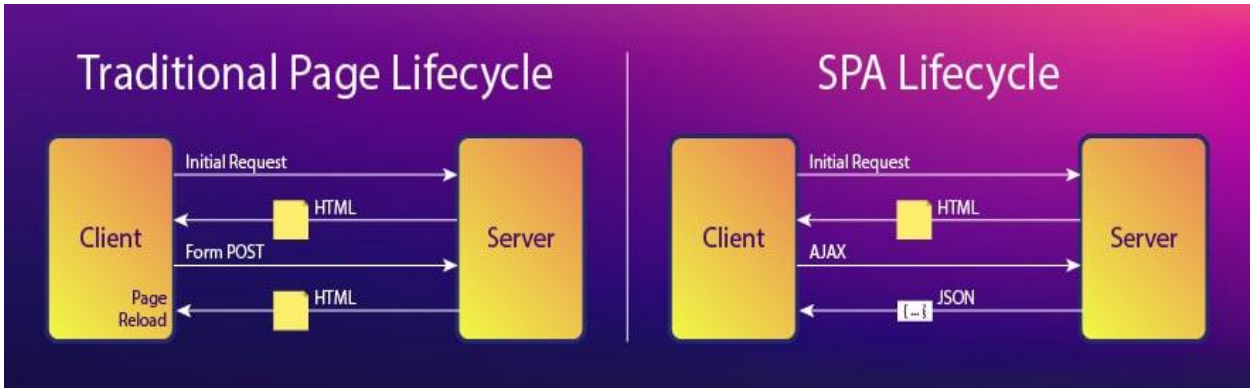


Рис. 2.1 Модель взаємодії клієнта з сервером двох типів

Слід зазначити, що з огляду на платформу у якій вестиметься розробка, створення багато-сторінкових додатків з використанням сучасних середовищ розробки, не є можливим, так як доступ до ресурсів конфігурується всередині платформи, таким чином для розробки проекту зручніше використовувати SPA підхід.

Серед зазначених на початку розділу бібліотек, для розробки SPA нам більше підходить React або Vue, так як даних бібліотеках реалізованих компонентний підхід для створення такого типу додатку, також навідмінну від Anglar, ці бібліотеки є мінімалістичними, хоча забезпечують майже всім необхідним. Найпопулярнішим серед розробників на сервісі Github, є React, через просту структуру додатків, бібліотека має ‘малий поріг входу’, тобто вивчивши основи, легко почати розробку власних додатків. Таким чином на даний момент серед розробників на бібліотеці React сформувалась велика комуна, на різних платформах є велика кількість навчальних матеріалів.[9][2][7]

2.1.2 Допоміжні бібліотеки

Бібліотека react-router-dom

З огляду на те, що веб додаток у даній роботі буде SPA типу, маршрутизація буде проводитись зі сторони клієнта. В даній роботі використовується бібліотека “react-router-dom”, так як звичайна маршрутизація React не працюватиме з ServiceNow, оскільки ми не маємо повного контролю шляху до ресурсів, з огляду на це лишається тільки клас HashRouter, надалі хеш-маршрутизатор.

Так як сервер ігнорує символ решітки ‘#’ та все що іде після нього в адресному рядку в браузері, хеш-маршрутизатор використовує це для побудови маршрутів до різних частин додатку, наприклад «https://www.<домен >.com/#<головна>», таким чином сервер у відповідь на зміну шляху після решітки не повертає відповідь з кодом 404, що означає – сторінку з даним шляхом не знайдено, а рахує, що маршрут не змінився. В цей час клієнтська частина додатку слідкує за зміною шляху, та повертає відповідний компонент (детальніше див. підрозділ 2.2.1).[15]

Бібліотека Axios

Axios - це широко відома JavaScript-бібліотека. Вона являє собою HTTP-клієнт, заснований на Проміс і призначений для браузерів і для Node.js. У жовтні 2019 року пакет Axios був завантажений з npm 25 мільйонів разів. Бібліотека

набрала таку популярність через те, що раніше вбудований в браузер стандартний інтерфейс XML HTTP Request, був незручний та його шаблонний код займав у кілька разів більше рядків. Проте у 2015 році був випущений новий вбудований в браузер інтерфейс Fetch API, який в певній мірі компенсує недолік XHLHttpRequest див. приклад нижче:

1) XMLHttpRequest

```
let xhr = new XMLHttpRequest();
xhr.open('GET', '/article/xmlhttprequest/example/json');
xhr.responseType = 'json'; xhr.send(); xhr.onload = function() {
    let responseObj = xhr.response;
    alert(responseObj.message);
};
```

2) Axios

```
axios.get("https://jsonplaceholder.typicode.com/todos/1")
.then(response => console.log("response", response.data))
```

3) Fetch

```
fetch('https://jsonplaceholder.typicode.com/todos/1')
.then(response => response.json())
.then(json => console.log(json))
```

Окрім компактності Axios, має змогу стежити за ходом вивантаження даних, що дозволяє користувачам вивантажувати на сервер фотографії або відеофайли, також дає змогу налаштувати всі запити чином, щоб автоматично разом з запитом передавати інформацію про авторизацію, а з огляду на те, що для доступу до ресурсів екземпляру потрібна авторизація, це у невеликій мірі зменшує написання шаблонного коду.**[3]**

Препроцесор SASS

Для стилізації сайту в даному проекті використовується препроцесор SASS – це розширення CSS, яке надає можливість використовувати змінні, міксини, імпорт файлів і багато іншого, все з повністю сумісним з CSS синтаксисом. Sass допомагає зберігати величезні таблиці стилів добре організованими, а невеликим

стилям працювати швидко. SASS дає можливість вкладати селектори в середину інших селекторів, що дозволяє компонувати класи для повторного використання, що значно зменшить кількість коду. Можливість вкладати селекторы дає можливість зручно використовувати БЕМ (блок-елемент-модифікатор) методологію, що дозволяє розробникам легше розуміти та орієнтуватися в коді. З можливістю імпорту файлів SASS, дозволяє поділити код на різні файли, які в результаті імпортуються в один файл, який після компіляції перетворюється в звичайний index.css файл. Таким чином стилі веб додатку мають зручну структуру, як файлів, так і самого коду в файлах.[14]

Рис. 2.2 Порівняння CSS та SASS.

2.1.3 Інші інструменти

VS Code

В якості текстового редактору у даній роботі буде використовуватись VS Code, це редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кроссплатформенной розробки веб-і хмарних додатків.

ESLint

Також для написання коду використовується ESLint - це інструмент статичного аналізу коду для виявлення проблемних шаблонів, виявлених в коді JavaScript. Він був створений Ніколасом С. Закасом в 2013 році. Правила в ESLint можна налаштовувати, також можна визначати свої правила, таким чином редактор буде вказувати на стилістичні та синтаксичні помилки, в результаті

чого код легше читати та розуміти, що допомагає розробці в команді або доповненню власного додатку в майбутньому.

NPM

NPM - це менеджер пакетів, пакетом називається будь-який модуль або бібліотека, що використовується NodeJS додатками. Це схоже на Maven для Java або Composer для PHP. Існує два основних інтерфейса для взаємодії: сайт NPM виступає, як відкритий API для завантаження пакетів і набір команд командного рядка за управління. Кожен NodeJS додаток має в кореневій папці файл де і записуються всі дані про додаток, тобто назва, версія, пакети, що використовуються, їх версії тощо. Таким чином NPM дозволяє переносити додатки з комп'ютера на комп'ютер без модулів. Для того щоб встановити всі залежності достатньо прописати команду "npm install" і тоді всі залежності будуть збережені на локалькому носії.

Webpack

Webpack – це постачальник модулів. Іншими словами, Webpack бере купу різних активів або файлів (наприклад, CSS, JS, SASS, JPG, SVG, PNG ...) і об'єднує їх у групи, окремий пакет для кожного типу див рис. 2.1. В даній роботі Webpack збирає всі компоненти проекту в кінцеву структуровану зв'язку файлів та папок (детальний алгоритм див. підрозділ 2.1.4).

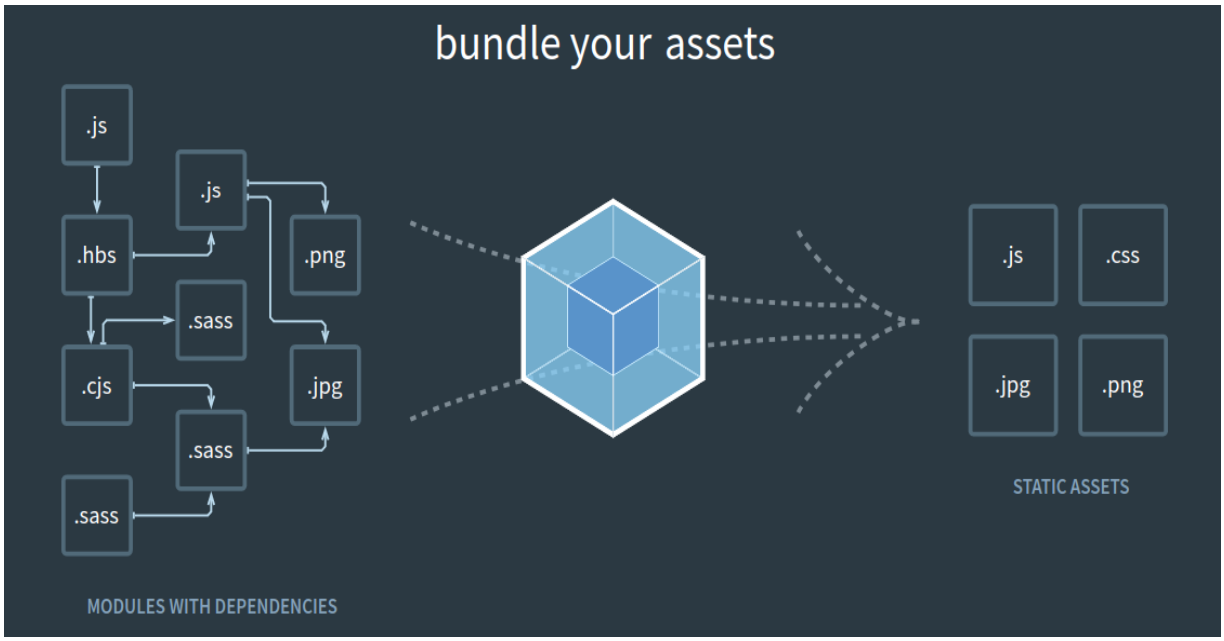


Рис. 2.3 Схема конвертації модулів у статичні файли[11]

2.1.4 Алгоритм розгортання програмного додатку

Розгортання програмного забезпечення (Розгортання ПЗ, англ. *Software deployment*) – це усі дії, що роблять програмну систему готовою до використання. Даний процес є частиною життєвого циклу програмного забезпечення. В контексті ServiceNow спочатку потрібно конвертувати додаток, щоб завантажувати його на платформі, в даній роботі буде використаний алгоритм запропонований Andrew Pishchulin в своїй статті “React in ServiceNow application (advanced)”.

Алгоритм наступний:

1. Завантажити пустий проект з базовими конфігураціями на власний комп’ютер [https://github.com/elinsoftware/sn-react]
2. Відкривши проект у зручному текстовому редакторі, потрібно змінити базові налаштування на свої дані, якими є, URL адреса, логін та пароль до власного екземпляру ServiceNow.
3. Далі потрібно встановити додаток проміжний додаток на платформу завантаживши update set, з github проекту, у якому находяться UI Page та три записи Scripted REST API див. підрозділ 2.3.
4. Зробивши певні зміни у проекті в текстовому редакторі, його потрібно конвертувати у мінімізований проект з кількох файлів, робимо це за допомогою двох команд в командному рядку.
   4.1 З допомогою команди “npm run build”, мы видаляємо папку з назвою dist з її вмістом (при наявності), потім створюємо нову папку з назвою “dist”, та заповнюємо її головним файлом index.html, та трьома папками з файлами: javascript (для зберігання .js файлів), images (зберігаються картинки форматів .jpeg .png та ін.) та assets (зберігаються будь-які інші файли, які використовуються в додатку, наприклад .svg).

4.2 Далі з допомогою команди “npm run postbuild”, мы змінюємо основний html файл, замінюючи певні теги на jelly теги, та додаємо логіку авторизації.

5. Виконавши вище вказані команди, ми маємо готовий проект, який потрібно завантажити в систему.

   5.1 Копіюємо вміст файлу index.html в папці dist, та вставляємо його в запис в плаформі UI Page, саме в поле html та зберігаємо даний запис.

   5.2 Далі потрібно додати в закріплення вміст папок javascript, images та assets до відповідних записів REST API, створених раніше.

6. Зробивши вище вказані дії, за посиланням в полі endpoint запису UI Page, буде доступний додаток.[6]

2.2 Розробка клієнтської частини

2.2.1 Базові поняття

В попередньому підрозділі ми обрали React, як основний інструмент для розробки клієнтської частини, розробка додатків на React передбачає наступні підходи:

- Компонентний підхід – компоненти дозволяють розділити інтерфейси на незалежні частини, якими легше маніпулювати. Можна скласти разом і використовувати кілька раз.
- Props, або Властивості використовуються для взаємодії компонентів один з одним, передаються схожим на аргументи функції параметри.
- Стан компоненту – опис в середині компоненту його станів, та методів взаємодії з ним, і на основі стану компонент буде змінюватись.
- Композиція – компоненти можуть посилатися на інші компоненти, можливо використати одну і ту же абстракцію – компоненти – на будь-якому рівні додатку. Неважливо, це

кнопка або цілий екран: всі вони, як правило, представляють окремі компоненти в React-додатках.

- JSX – це технологія за якою компоненти виглядають як звичайні html (<Element />) всередині javascript коду, відмінність в тому, що деякі зарезервовані в javascript слова, типу “class”, замінені.

Приклад компоненту:

```
function Welcome(props) {
    return <h1>Привет, {props.name}</h1>;
}
```

Таким чином React додаток виглядає як, зв’язка файлів, в яких описані компоненти, що повертають html-подібні теги в деревовидній формі та містять у собі методи для взаємодії з цими тегами. Ці файли в результаті компіляції з допомогою Babel, перетворюються в звичайний javascript код, який розуміє браузер.[13]

2.2.2 Основні компоненти

Компоненти в даній роботі розділені на два типи:

- Функціональний компонент, який є окремою одиницею функціональності, наприклад меню, список, кнопка;
- Компонент-сторінка, який слугує обгорткою в якій компоненти об’єднуються в композицію і працюють як одна система, наприклад список інцидентів з посиланням та контрольна панель з фільтрами.

Основні функціональні компоненти:

Компоненти підтипу List

Це Компоненти, які приймаються список об'єктів, та з допомогою вбудованого методу map повертається масив відповідних компонентів, серед них:

- IncidentList
- RequestItemList
- KnoledgeBaseList
- AssetList
- ApprovalList
- TaskList

Компоненти підтипу елемент

Компонент, посилання на окрему сторінку відповідного компонента з розгорнутою інформацією про об’єкт, серед них:

- Incident
- RequestItem
- KnoledgeArticle
- Task
- Approval
- Asset

Компонент DropDownElement

Компонент використовується як окремий елемент на контрольних панелях з фільтрами. Використовується на сторінках зі списками, де потенційно може бути декілька сторінок, в якості параметрів приймає назву таблиці ServiceNow, назву поля типу choice, що є аналогом тегу select та функцію зворотного виклику, яка виконується кожен раз коли значення поля змінюється.

Компонент RequestAction

Компонент-форма, який відкривається в модальному вікні для заповнення деталей запиту, наприклад запит до додаткове обладнання, запит за питання або відпустку тощо.

Відповідно до ролі користувача йому буде доступний різний набір можливий запитів.

Компонент RequestActionPanel

Компонент-Панель на сторінці зі списком проведених користувачем запитів, являє собою вертикальний список, компонентів кнопок, при натисканні на яку відкриється відповідний елемент RequestAction.

Компонент Navigation

Компонент, який знаходиться зверху кожної сторінки і містить зліва-направо посилання на головну сторінку, пошуковий рядок, та три елементи-посилання на список інцидентів, запитів та персональну сторінку з налаштуваннями для користувача.

Компонент Footer

Компонент, який завжди знаходиться знизу сторінки і містить в собі список з посиланням на побічні сторінки типу: контакти, правила копірайт.

Компоненти типу сторінка

HomePage

Сторінка містить в собі елементи навігації по усьому порталу через компоненти-посилання, які складаються з картинки та надпису.

Компоненти-сторінки підтипу список

Це наступні компоненти списки:

- IncidentsPage
- RequestItemsPage
- KnoledgeBasePage
- AssetsPage
- ApprovalsPage
- TasksPage

Вони складаються зі списку елементів-посилань та фільтрів до них, да зручного пошуку. Компонент-сторінка RequestItemsPage, окрім вище перерахованих компонентів містить в собі панель з доступним для користовача RequestAction, які знаходяться на панелі

Компоненти-сторінки підтипу деталі

Це наступні компоненти:

- IncidentPage
- RequestItemPage
- KnoledgeArticlePage
- AssetPage
- ApprovalPage
- TaskPage

Відповідно до кожного компоненту сторінки містять в собі розгорнуту інформацію про певний об’єкт, та контрольну панель з певними діями доступними в концексті об’єкту, наприклад, сторінка IncidentPage містить дію ‘Resolve’, сторінка TaskPage ‘Close’, сторінка ApprovalPage ‘Approve’ та ‘Reject’, сторінка RequestItemPage ‘Complete’ та ‘Cancel’ тощо.

UserPage

Сторінка користувача містить можливі відповідно до ролі користувача дії з персональною інформацією та можливими конфігураціями, наприклад зміна теми тощо.

2.3 Інструменти для розробки та створення серверної частини

2.3.1 Серверна розробка в ServiceNow

В підрозділі 1.7 були визначені більшість серверних типів скриптів серед них в даній роботі ми будемо використовувати три:

- Business rule, надалі «правило»
- Scripted REST API, «апі»
- Script include, надалі «скріпт»

У цій зв'язці «апі», буде отримувати запит з клієнтської частини, та на основі цього викликати певний «скріпт» для обробки або повернення даних. «Правило» буде виконувати певні дії у відповіть на певні зміни в таблицях.

2.3.2 Власні та системні REST API

В системі попри можливість створювати власні «апі», є «апі» для отримання будь-яких даних в системі, серед них виділимо Table API, що дозволяє отримати дані з будь-якої таблиці, недоліком даного «апі» в тому, що воно повертає дані за замовчуванням, тобто неможна вказати, які поля та яке значення потрібне. Проте він, як готове рішення покриє частину логіки проекту, серед них загальна інформація для сторінок з відображенням списку елементів.

Також потреба у створенні власних «апі» виникає в тому, що більшість даних мають поля типу «посилання» на інші таблиці, таким чином набагато зручніше одним запитом до серверу отримати дані, про декілька записів в різних таблицях, аніж для кожного поля с типом «посилання» робити новий запит. Таким чином можна мінімізувати кількість звернень до серверу і додаток буде працювати швидше.

Серед власних «апі» виділимо наступні:

UserApi

Використовується для отримання детальних даних про користувача, що використовується як контекст авторизації, щоб ідентифікувати користувача, таким чином відображається ім’я та фото користувача у верхньому правому кутку сторінки, та додаток розуміє набір ролей даного користувача. Також використовується для відображення інформації на сторінці ***UserPage*** та відповідно до конфігурацій користувача відображається тема додатку.

IncidentApi, RequestApi, KnoledgeBaseApi

Використовуються до отпримання деталей про записи та для здійснення певних дій з цими записами. Відповідно до «апі» можна або створити запис,

редагувати або видалити запис. Кількість можливих дій залежить від типу запису, виключення є Request Api, так як там опрацьовуються запити.

RequestActionApi

Це «апі» має призначення, обробляти запити з компоненту «RequestActionPanel», тобто дії по замовленню обладнання та будь-які інші можливі користувацькі запити.

2.3.3 Основні серверні рішення додатку

В даному підрозділі описаний функціонал серверних скриптів, які виконують логіку взаємодії з записами в таблицях. Розглянемо скрипти які виконуються з допомогою апі:

EmployeeManagementCart

Проміжний скрипт який повертає об’єкт “Cart”, що допомагаю у створенні користувацького запиту.

EmployeeManagementCardHelper

Скрипт, який на основі користувацького запиту (типу запиту та його параметри) обробляє та заповнює об’єкт “Cart”, відповідно до заповненої інформації у запиті на сторонці RequestsPage, створює новий запис. Скрипт має метод для обробки параметрів та запису їх у відповідні змінні у Request Item, та методи для обробки дій, які знаходяться на сторінці деталей конкретного запиту.

EmployeeManagementUtils

Скрипт, який містить методі для взаємодії з даними загального призначення, серед методів в ньому є функції для роботи з датами (додати певну кількість місяців до дати), записи (зібрати всі записи типу коментар в об’єкт та повернути).

EmployeeManagementUserDataHelper

Скрипт містить у собі методи для взаємодії з записами таблиці користувачів, у тому числі скрипт повертає потрібні дані за запитом, наприклад, дані про поточного користувача в системі, дані конкретного користувача за його ідентифікаційним номером, шлях до фото користувача в системі тощо.

EmployeeManagementIncidentHelper

Скрипт для взаємодії з записами в таблиці Інцидентів, серед методів містить наступні – додавання коментарів, призначення відповідального користувача, зміна статусу тощо. В скрипті є функції для обробки дій “Resolve” “Cancel”, “Create Child Incident” тощо.

EmployeeManagementKnoledgeBaseHelper

Скрипт для функціонального та «розумного» пошуку серед статей в базі знань, зміни, створення нових статей та можливістю додавання статті до обраних для певного користувача тощо.

EmployeeManagementTaskHelper

Скрипт функціонально схожий до EmployeeManagementIncidentHelper за невеликими відмінностями відповідно до полів в таблиці, та функціональних дій на сторінці з задачами, наприклад “Close”.

EmployeeManagementApprovalHelper

Скрипт містить методи для обробки дій на сторінці таких, як “Approve” та “Reject”, функціонал створення записів, відповідно до ролі користувача.

EmployeeManagementAssetHelper

Скрипт для маніпуляціями з активами користувачів, зміна даних. Містить метод для обробки дії “Reassign Asset” та інше.

Також використовуються скрипти «правила» (або Business rule), які як відомо з першого розділу виконуються відповідно до заданих змін в таблицях, серед них:

EM Create User Configuration

Правило, що створює сутність User Configuration, зі значеннями за замовчування, коли в системі створюється новий користувач.

EM Delegate All Responsibilities

Правило, що призначає всі записи в таблицях у яких є поле типу «призначений користувач», на користувача, що находиться в списку “Delegate”, у випадку коли користувач стає неактивний.

2.4 Огляд основних сутностей додатку

2.4.1 Системні сутності

В даній роботі здебільшого використовуються системні сутності, також деякі системні таблиці були доповнені новими полями. Використовуються наступні таблиці:

User (sys_user) – таблиця, що зберігає дані про користувача.

Табл. 2.1 – Таблиця користувачів.

Назва поля	Тип поля	Додатково
First name	String	Ім’я
Last name	String	Прізвище
Manager	Reference	Посилання на таблицю User
User ID	String	Логін користувача
Email	String	Електронна пошта
State	Choice	Поточний стан користувача. Може бути одним з наступних: Active, On vacation, Inactive, Temporary inactive
Password	Password	Поле в якому зберігається хеш пароля

Incident (incident) – зберігає записи інцидентів

Табл. 2.2 – Таблиця інцидентів.

Number	String	Унікальний номер інциденту
Caller	Reference	Посилання на таблицю User, автоматично заповнюється користувачем потомним користувачем при створенні
Category	Choice	Один з наступних варіантів: Injuri/Help, Software, Hardware, Network, Database
State	Choice	Один з наступних варіантів: Open, Work in Progress, Close Incomplete, Close Complete Pending
Priority	Choice	Один з наступних варіантів: 1, 2, 3, 4
Assigned to	Reference	Посилання на таблицю
Short Description	String	Максимальна довжина 40 символів
Description	String	Максимальна довжина 200 символів

Task (task) – таблиця для зберігання записів задач.

Табл. 2.3 – Таблиця задач.

Number	String	Унікальний номер задачі
Assigned to	Reference	Посилання на таблицю User
Priority	Choice	Один з наступних варіантів: 1, 2, 3, 4
State	Choice	Один з наступних варіантів: Open, Work in Progress, Close Incomplete, Close Complete Pending
Short description	String	Максимальна довжина 40 символів
Description	String	Максимальна довжина 200 символів
Task type	Choice	Change, Catalog, Change Request

Catalog Item (sc_cat_item) – таблиця для зберігання елементів каталога

Табл. 2.4 – Таблиця елементів каталогу.

Name	String	Назва предмету
Catalog	Reference	Посилання на таблицю Catalog
Category	Reference	Посилання на таблицю Category
Short Description	String	Максимально 40 символів
Description	String	Максимально 200 символів
Workflow	Reference	Посилання на таблицю Workflow
Icon	Photo	Фото елементу
Price	Currency	Ціна у заданій валюті

Сутність елементів каталогу див. Табл. 2.4, представляє собою шаблони для запитів користувача, є одним з основних функціоналів платформи, його суть в тому що до кожного такого шаблону потрібно заповнити поле Workflow, що вказане вище. Workflow – це основна ідея автоматизації платформи, він складається з елементів, які виконуються в залежності від певних дій користувачів, та у напів-автоматичному режимі, виконують задані операції див. рис. 2.2, Приклад Workflow.

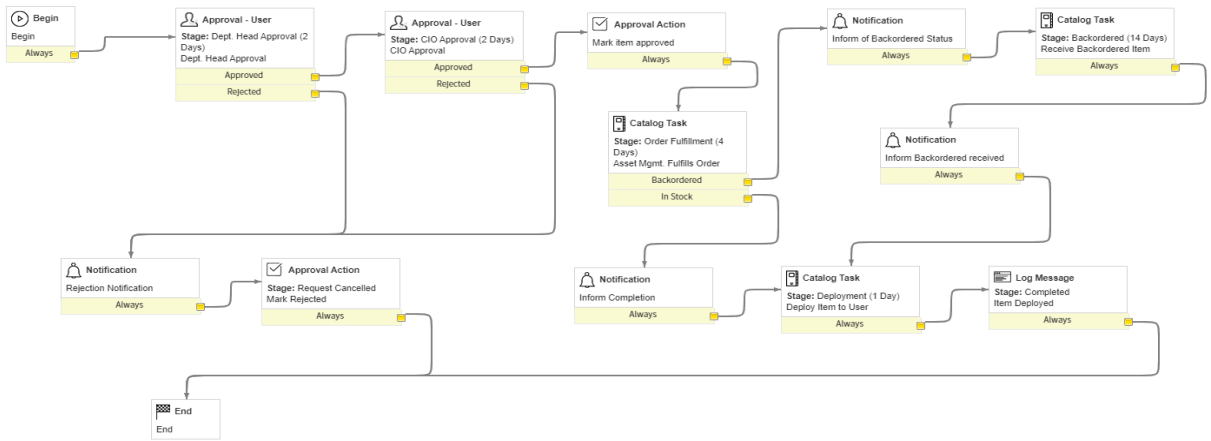


Рис. 2.3 Приклад Workflow

Окрім Workflow, ще одна річ робить Catalog Item гнучкими, це Variables, або змінні, це окрема таблиця з записами, де власне записи схожі на записи полів

в таблицях, тому що мають поля назва та тип. В сутності змінні, це можливість робити запити з певними параметрами, наприклад запис в таблиці Catalog Item з назвою “Samsung S8” може мати змінну з назвою об’єм памяті, таким чином в залежності від об’єму буде змінюватись ціна сматрфону.

Request Item (sc_req_item) – зберігає дані про запити деталей каталога

Табл. 2.5 – Таблиця запитів елементів.

Number	String	Унікальний номер запиту
Item	Reference	Посилання на таблицю Catalog Item
Requested for	Reference	Посилання на таблицю User
Due Date	Date	Дата потенційного завершення обробки запиту
Stage	Choice	Частіше один з наступних варіантів: Approval, Fulfilment, Delivery, Closed Complete
State	Choice	Один з наступних варіантів: Open, Work in Progress, Close Incomplete, Close Complete Pending

Approval (sysapproval_approver) – зберігає записи схвалень.

Табл. 2.6 – Таблиця схвалень

Approver	Reference	Посилання на таблицю User, користувач, що має Схвалити Запис
State	Choice	Одне з наступних значень Requested, Approved, Rejected, Pending
Approving	Document Id	sys_id запису з таблиці, що потребує схвалення

.

2.4.2 Власні сутності

Окрім системних таблиць було створено кілька власних таблиць, серед них:

User Configuration

Табл. 2.7 – Таблиця користувацьких налаштувань.

User	Reference	Посилання на таблицю User
Theme	Choice	Один з наступних: Pink, Blue, Violet, Green

Action Configuration

Табл. 2.8 – Таблиця налаштувань запитів.

Name	String	Ім’я
Action name	String	Значення для запиту
Available for	String	Один з наступних: any,admin, hr, itil.
Item	Reference	Посилання на таблицю Catalog Item

2.5 Ролі користувачів

В системі ServiceNow за надання можливостей певному користувачу відповідають ролі за якими стоїть окрема таблиця «sys_user_role». В даній роботі підтримується така модель. Кожного разу, при завантаженні додатку, виконується запит до User Api (описаний в підрозділі 2.3.1), де сервер повертає дані при поточного користувача (поточний користувач визначається системним методом платформи), зокрема список ролей користувача, та його запис в таблиці User Configuration, що містить інформацію про налаштування порталу поточного користувача.

В додатку використовуються дві системні ролі:

- ***ITIL*** – має повноваження змінювати користувацькі записи в таблицях Incident, Task, Request. Користувачі з даною роллю виконують функцію вирішення користувальких питань, та обробку їх запитів.

- ***Admin*** – це системна роль яка має найбільші повноваження в системі (окрім, security admin) та включає в себе всі можливості, найчастіше цю роль дають головним адміністраторам або розробникам.

Окрім системних використовуються дві власні ролі, серед них:

- **HR** – роль, яка виконує певні запити користувача, створена загалом для допомоги користувачам, на вирішення не технічних питань.
- **End-user** – абстрактна роль, рахується що користувач має цю роль, якщо не має ні однієї з вище перечислених.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

У даному розділі було продемонстровано деталі розробки UI частини було використано найпопулярнішу на даний час бібліотеку React. Був продемонстрований алгоритм розгортання дотатку на платформі ServiceNow, у ході чого був використаний інструмент Webpack для мініфікації та генерування JavaScript коду у версії, доступної для платформи.

Було продемонстровано на описано дерево компонентів користувацької частини, у ході чого вдалося класифікувати більшість на чотири типи:

- Компонент-сторінка зі списком.
- Компонент-сторінка з деталями.
- Компонент-список.
- Компонент-картка.

По даній класифікації та виключенням був описаний функціонал кожної сторінки, який вийшов логічно-зрозумілий та простий у використанні.

Були описані серверні скрипти, що виконуються на платформі, у відповідь на запит до API платформи.

Також було розглянуто сутності, що використовують у даній роботі, вони були поділені на два підтипи, системні та власні. Здебільшого в даній роботи використовувалась системні ролі, так як базові ITIL інструменти уже вбудовані в системі.

В кінці були розглянуті ролі в системі, в залежності від якої користувач в системі має відповідні права.

РОЗДІЛ 3
ДЕМОНСТРАЦІЯ РОБОЧИХ ПРОЦЕСІВ

3.1 Навігація та інструкція по використанню.

Посилання на порталі.

Портал у даній роботі реалізований, як окрема користувацька сторінка, щоб перейти на неї, потрібно:

1) Авторизуватись до екземпляру ServiceNow, після чого відкриється головна сторінка.
2) Зліва в пошуковому рядку, написати ‘My Portal’.
3) Натиснути на модуль ‘My Portal’, що знаходиться у додатку ‘Employee Management’.

Після виконання вказівок вище, відкриється домашня сторінка порталу, що зображена на рис. 3.1.

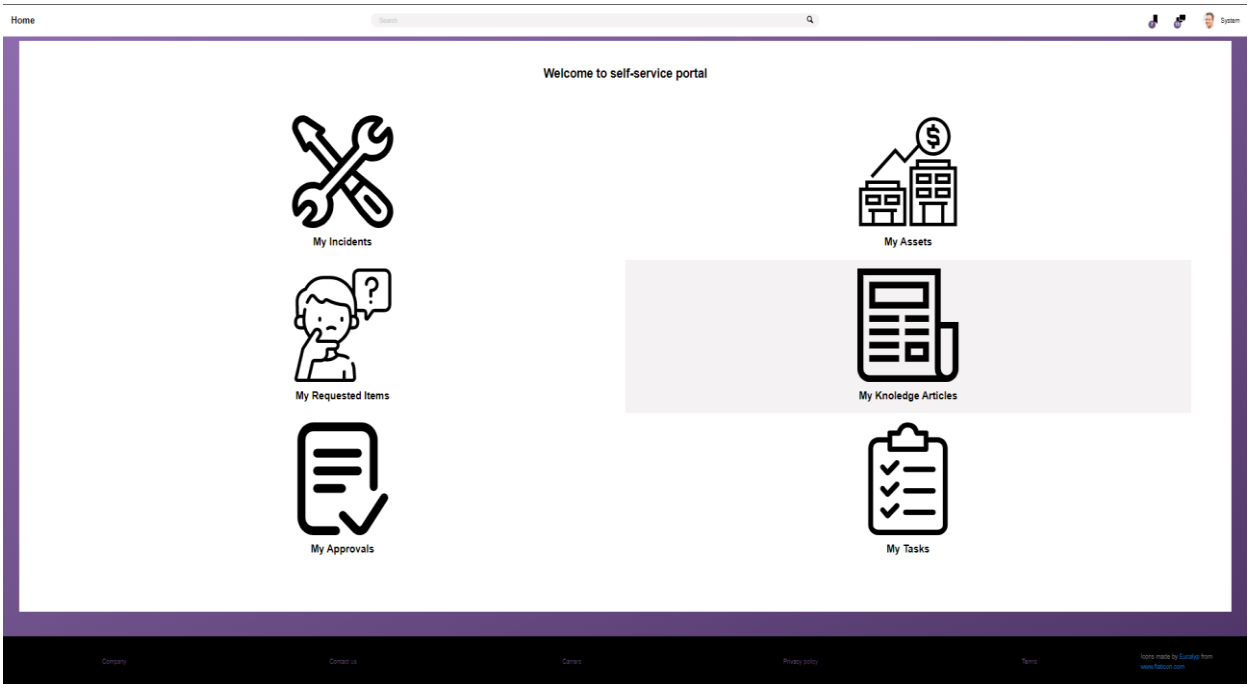


Рис. 3.1 – Домашня сторінка порталу

Для навігації на основну сторінку виведені посилання на 6 основних списків, це My Incidents, My Requested Items, My Approvals, My Tasks, My Knoledge Articles, My Assets. Варто зауважити, що залежно від ролі, користувачу буде доступна різна певна кількість посилань. Таким чином:

- Для користувача з роллю ‘admin’, доступні всі посилання.

- Для користувача з роллю ‘itil’, доступні всі окрім, My Knoledge Articles, My Assets.
- Для користувача з роллю ‘hr’, доступні всі окрім, My Incidents, My Tasks.
- Для користувача з роллю ‘end-user’ доступні всі окрім, My Approvals, My Tasks.

Якщо користувач спробує перейти на недоступну йому сторінку за посилання, його буде перенаправлено на головну сторінку. Окрім посилань на головній сторінці, для зручної навігації можна використовувати верхню панель див. рис. 3.2.

Рис. 3.2 – Верхня панель порталу

На панелі є чотири елемента, серед них:

- Кнопка ‘Home’, при натисканні на яку, відкриється домашня сторінка.
- Пошуковий рядок, що при вводі запиту та натисканні клавіші ‘Enter’, перенаправить користувача на сторінку “My Knowledge Articles” з застосованим фільтром, по тексту статті.
- Два посилання на списки відповідно до ролі користувача, наприклад у ‘end-user’ це My Incidents, My Requested Items, також на цих посиланнях позначено кількість активних елементів відповідного списку, таким чином користувач, зайшовши на портал відразу побачить кількість відкритих інцидентів.
- Посилання на сторінку з інформацією про користувача, на посиланні зображена фотографія користувача та його ім’я.

Сторінки-списки.

На рисунку 3.3 зображено приклад сторінки-списку, на ній знаходиться панель з фільтрами, та список карток, с короткою інформацією про окремий запис.

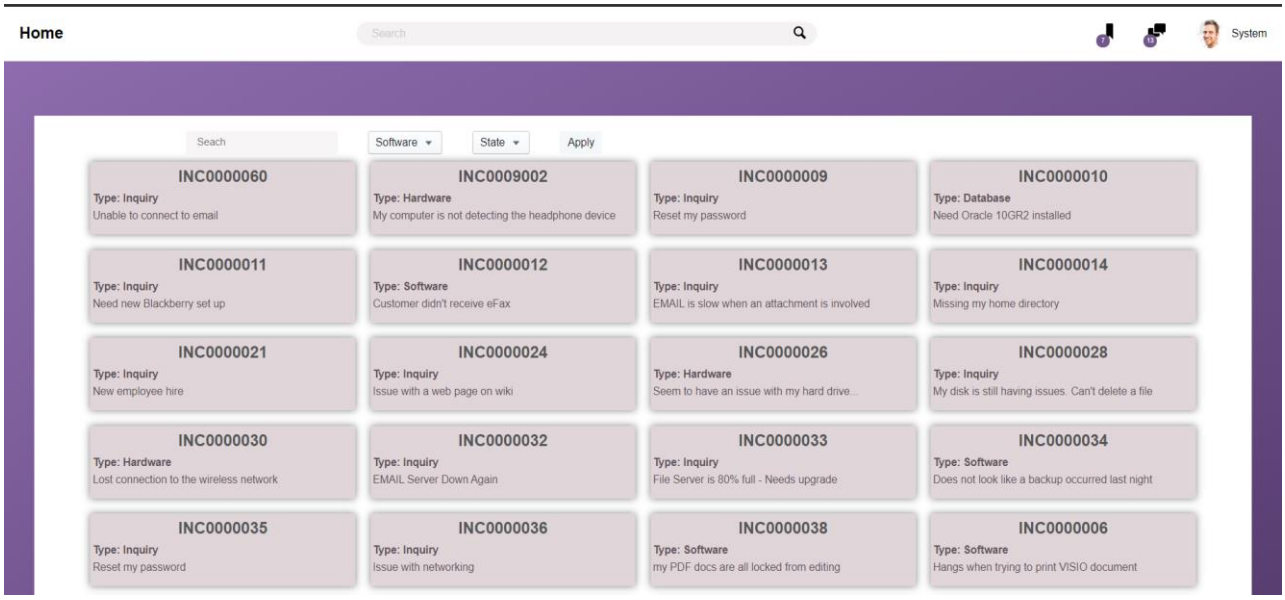


Рис. 3.3 Приклад сторінки списку

Вийнятком серед сторінок-списків – є сторінка My Requested Items (див. рис. 3.4), окрім фільтра та списку з записами на ній зображена панель з сервісами, які користувач може замовити. Відповідні сервіси доступні відповідним користувачам, за доступність відповідає конфігураційна таблиця Action Configuration, що описана в підрозділі 2.4.2.

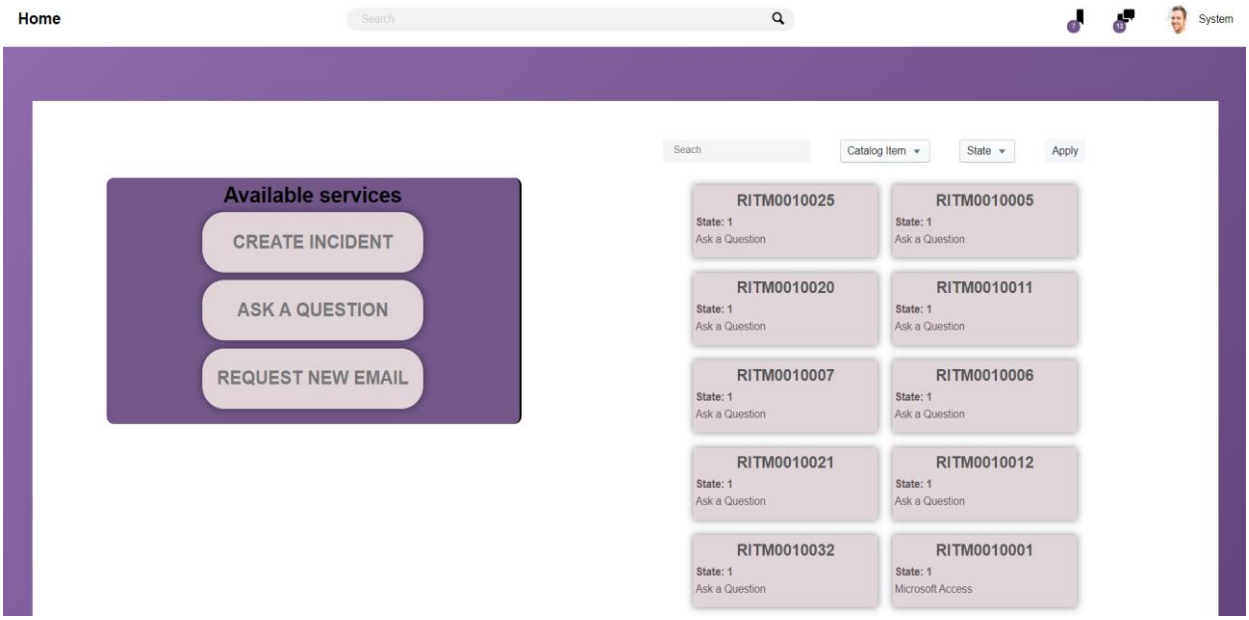


Рис. 3.4 – Сторінка My Requested Items

Сторінки записів

Для кожного запису є окрема сторінка, залежно від таблиці, сторінка наповнена різними деталями про запис та зв'язані з ним записи з інших таблиць, наприклад коментарі до запису. На рис. 3.5 зображений приклад сторінки-запису, у даному випадку це сторінка інциденту, вона складається з 4 компонентів:

- Заголовок, що складається з номеру інциденту та його скороченого опису.
- Блок з основною інформацією.
- Панель з кнопками, для здійснення операцій доступних для обраного запису.
- Панель з коментарями до записа та полем для вводу нового коментаря.

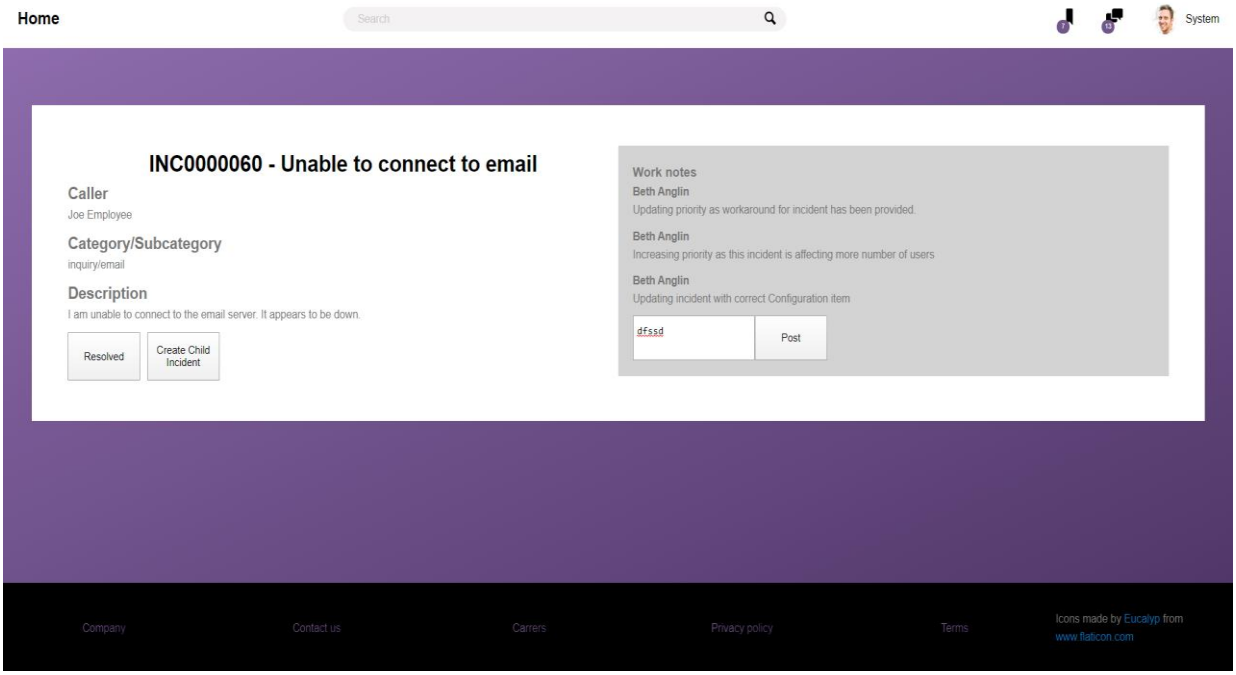


Рис. 3.5 – Приклад сторінки-запису

Пошук у базі знань

Як було зазначено вище, при вводі у пошуковий рядок, користувача буде направлено на сторінку зі статтями, з заданим фільтром при вводі, при цьому на

сторінці “My Knowledge Articles” пошуковий рядок у шапці сайту зникає, та рядок вводу копіюється у локальний рядок для пошуку, що знаходиться на панелі фільтрів.

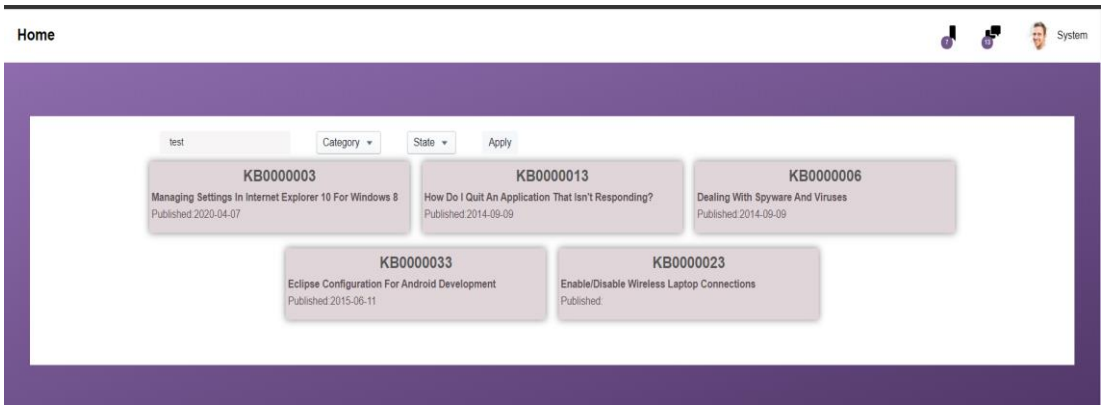


Рис. 3.6 – Приклад результату пошуку

3.2 Опис робочих процесів.

З попереднього розділу видно, що з інтерфейсі порталу реалізовано базові можливості шести ITIL інструментів, серед них:

Request Management

На сторінці ‘My Request Items’, у користувача є можливість створювати запити на доступні для нього сервіси, та на тій же сторінці відслідковувати їх процес оброки. Наприклад на рис. 3.7 зображене вікно запиту на питання.

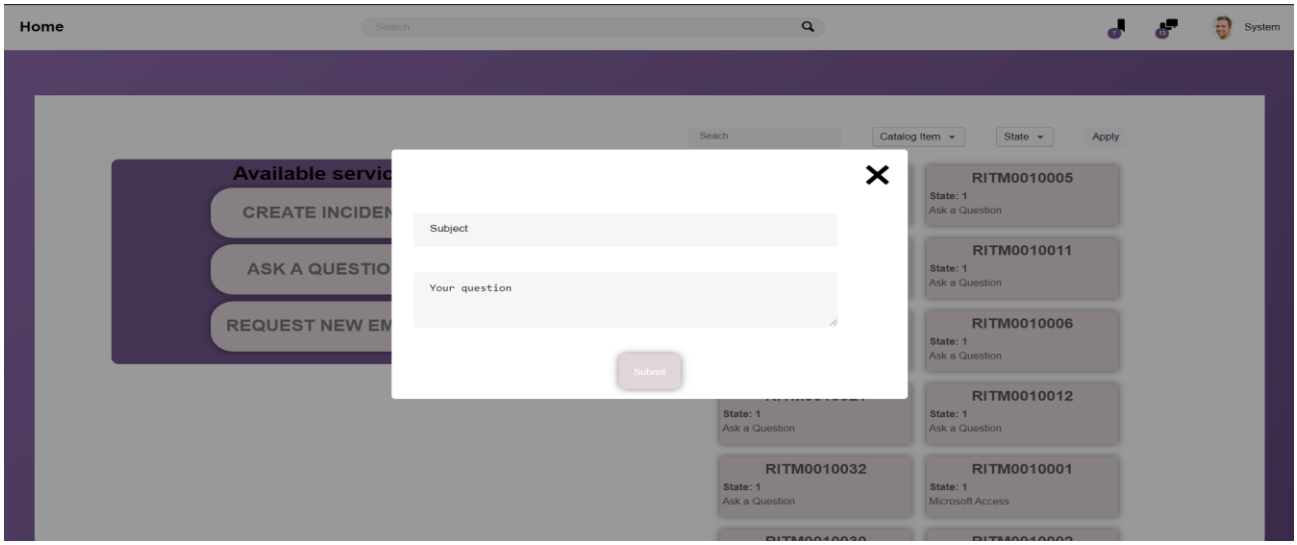


Рис. 3.7 – Приклад форми для запиту

Task Management

При обробці користувацьких запитів, можуть бути створені задачі на конкретних користувачів або груп, для вирішення певного етапу обробки,

наприклад, при замовлені програмного забезпечення або апаратури, буде створена задача на користувача систему, що відповідальний за замовлення.

Incident Management

В системі є можливість створювати нові інциденти та відслідковувати їх обробку.

Approval Management

При створенні певних запитів можуть бути створені записи схвалення. Наприклад, при запиті користувача на сервіс, що коштує більше ніж $1000, буде створено запис схвалення, де за схвалення буде відповідати менеджер користувача.

Asset Management

Деякі сервіси при обробці створюють активи для користувачів, серед них може бути техніка або віртуальні активи такі як, сервер або лінія мобільного зв'язку. Для менеджменту активів в порталі відведена окрема сторінка, в якій можна дізнатися про стан активу або виконати певні операції, наприклад віддати актив у розпорядження іншому користувачеві.

Knowledge Base

Важливою функцією порталу є можливість знайти відповідь на своє запитання зв'язане з робочим процесом. База знань є інструментом для вирішення частини питань, щоб зменшити навантаження на відділ підтримки у компанії, таким чином підійшовши відповідально до наповнення бази знань, створивши статті з заготовленими відповідями на найпоширеніші питання, можливо мінімізувати час вирішення питань. Потрапити у базу знань у даній роботі можна двома способами:

1) Натиснувши на посилання 'My Knowledge Articles' на домашній сторінці, після чого відкриється список з усіма статтями.
2) Ввівши запит в поле 'Search' на верхній панелі додатку та натиснувши клавішу 'Enter'. У даному випадку відкриється сторінка зі списком 'My Knowledge Articles' з застосованим фільтром, що буде шукати

статті по короткому опису або тілу статті, таким чином можна шукати відповіді на запитання швидко з будь-якої сторінки.

3.3 Перегляд додаткових можливостей.

3.3.1 Розумний пошук

На сторінках зі списками записів, на панелі фільтрів, в рядку ‘Search’, реалізовано розумний пошук. Ввівши рядок у форматі <назва_рядку>:<значення>, можна шукати по будь-якому полю в таблиці, наприклад, на сторінці ‘My Incidents’ ввівши запит ‘short_description:issue’, будуть виведені усі записи, у яких вказане поле містить вказане значення див. Рис. 3.8.

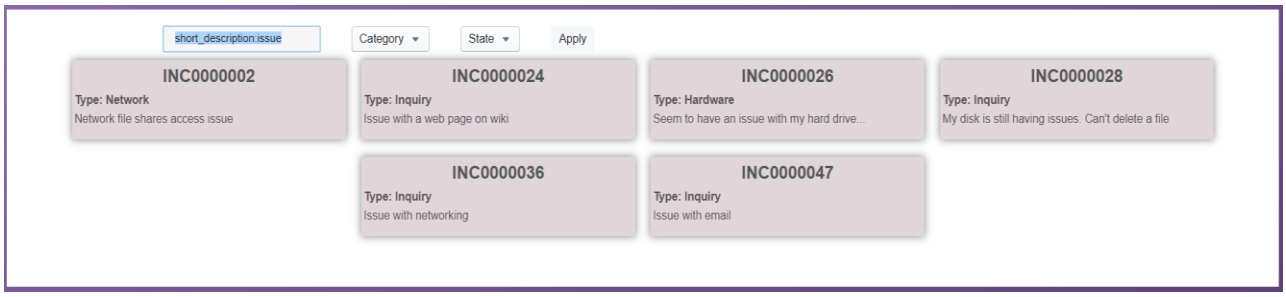


Рис. 3.8 – Приклад використання розумного пошуку.

Дана можливість направлена на допомогу досвідченим користувачам системи у швидкому пошуку потрібних даних, для користувачів незнайомих з платформою ввід буде шукати записи по-замовчування за полем, що є “display_value”, для відповідної таблиці, що конфігурується у системі.

3.3.2 Персоналізація

На сторінці користувача доступна базова інформація про користувача, та випадаючий список з можливими темами додатку див рис. 3.9. На вибір є чотири варіанта:

- Blue
- Pink
- Violet
- Green

Обравши тему та натиснувши ‘Apply‘ буде створений запит на зміну теми у поточного користувача, що зберігається у таблиці ‘User Configuration’ та перезавантажиться сторінка, щоб застосувати зміни.

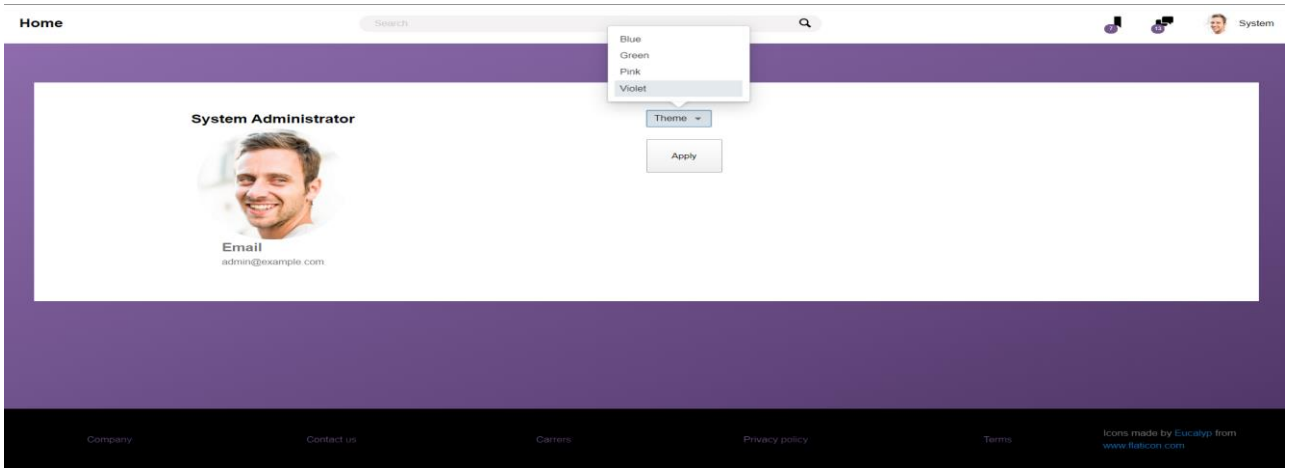


Рис. 3.9 Сторінка користувача

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

В даному розділі було розглянуто усі основні процеси, які були реалізовані у додатку та інтерфейси, які в них задіяні. У порівнянні з інтерфейсами платформи, де користувач може мати велику кількість додатків в навігаторі та на багатьох формах, що є персональними сторінками окремих записів, майже завжди розміщається багато полів, що значно ускладнює процес пошуку. Також велика кількість полів на формі, які не використовуються, є зайвими для більшості користувачів. На порталі розробленому у даній роботі на сторінки записів виведена основна інформація.

Були описані основні процеси, що можливо виконати в даному додатку з по-кроковим описом інтерфейсів, що використовуються у даних процесах. Один з найважливіших інструментів, користувацькі запити, були винесені на сторінку з відповідною таблицею у якій відбувається процес обробки запитів.

Був описаний процес пошуку інформації у базі знань та вигляд головної сторінки додатку, для користувачів з різними ролями.

Також були продемонстровані додаткові можливості додатку, такі як:

- Розумний пошук, що дозволяє досвідченим користувачам платформи швидко шукати потрібні записи у таблиця по описаному полю.
- Персоналізація, що дозволяє користувачам обирати тему додатку.

ВИСНОВКИ

У даній роботі був використаний концептуально новий підхід, до створення користувацьких інтерфейсів на платформі ServiceNow. З використанням бібліотеки React було розроблено додаток у компонентному стилі, що являється кращим підходом, так як у платформі використовується багато блоків, що повторюються. Разом з системними API вийшом додаток, що працює швидше аніж стандартний портал самообслуговування.

У першому розділі була досліджена бібліотека ITIL та інструменти, що її реалізовують. Були порівняні різні системи, що реалізують цю бібліотеку. Здебільшого їх відмінність була в різноманітності реалізованих інструментів та у ціні. Був проведений детальний огляд до можливості створення власних додатків на платформі ITSM, у ході чого було описано, як система виконує скрипти та як взаємодіє з даними. Серед усього іншого були описані підхід платформи до контролю доступом до даних та додаткові можливості платформи, що дозволяють інтегрувати її до використання у будь-якій компанії.

У другому розділі був детально описано процес розробки як серверної та і клієнтської частини. Як вже було вказано раніше для розробки клієнтської частини використовувалась бібліотека React та інструменти, які використовуються разом з вищевказаною бібліотекою, серед них Axios, React-Router,Webpack, NPM, ESLint. Був використаний препроцесор CSS, що називається SASS, з яким зручно структурувати стилі. У ході розділу був розглянутий підхід до створення серверної частини в ServiceNow, та усі скрипти, що виконують код на сервері. Після чого були описані усі компоненти та їх функціонал. Також були розглянуті основні сутності та можливості користувачів, що різнилися в залежності від їх ролі.

У третьому розділі був описаний результат роботи. Були розглянуті основні робочі процеси, що були реалізовані. Були продемонстровані приклади усіх інтерфейсів в додатку. Був продемонстрований приклад запиту “Ask a Question”. В кінці були описані додаткові можливості дотатку такі як, персоналызація та «розумний пошук».

В результаті роботи кінцевий користувач, а в контексті платформи, це робітник компанії, що використовує ServiceNow, за допомогою даного додатку може здійснювати доступ до основних ITIL інструментів, у зручному інтерфейсі з можливістю персоналізації та швидкого доступу.

В даній роботі був розроблений повноцінний додаток для платформи ServiceNow, новизна даної розробки в підході та технологіях, що використовувались для створення користувацької частини. Всі задачі поставлені перед початком виконання роботи були виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. 11 Best ITSM Tools (IT Service Management Software) In 2020 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/itsm-tools/.
2. Angular vs React vs Vue: Which Framework to Choose in 2020 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.
3. Axios или Fetch: чем пользоваться в 2019 году? [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/477286/.
4. ITSM — что это такое и с чего начать внедрение [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/company/it-guild/blog/453526/.
5. Mastering ServiceNow – Birmingham: Packt Publishing Ltd., 2015. – 570 с.
6. React in ServiceNow applications (advanced) [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://medium.com/@pishchulin/react-in-servicenow-applications-advanced-3e1966fbb817.
7. React или Angular или Vue.js — что выбрать? [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/476312/.
8. Self Service: как и зачем учить пользователей самостоятельности [Електронний ресурс] // ИТ Гильдия. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/company/it-guild/blog/341926/.
9. Single-page application vs. multiple-page application [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
10. The unique record identifier (sys_id) [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.servicenow.com/bundle/orlando-platform-

administration/page/administer/table-administration/concept/c_UniqueRecordIdentifier.html.

11. Webpack 4 — The Complete Guide [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://medium.com/better-programming/webpack-4-the-complete-guide-af1b1e2e3f7a.

12. What is ServiceNow and How it is Changing the Face of ITSM? [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.stemjar.com/what-is-servicenow-capabilities/.

13. Компоненты и пропсы [Електронний ресурс] – Режим доступу до ресурсу: https://ru.reactjs.org/docs/components-and-props.html.

14. Препроцессор Sass. Полное руководство и зачем он нужен [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://medium.com/@stasonmars/%D0%BF%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80-sass-%D0%BF%D0%BE%D0%BB%D0%BD%D0%BE%D0%B5-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-%D0%B8-%D0%B7%D0%B0%D1%87%D0%B5%D0%BC-%D0%BE%D0%BD-%D0%BD%D1%83%D0%B6%D0%B5%D0%BD-20fb638e29e3.

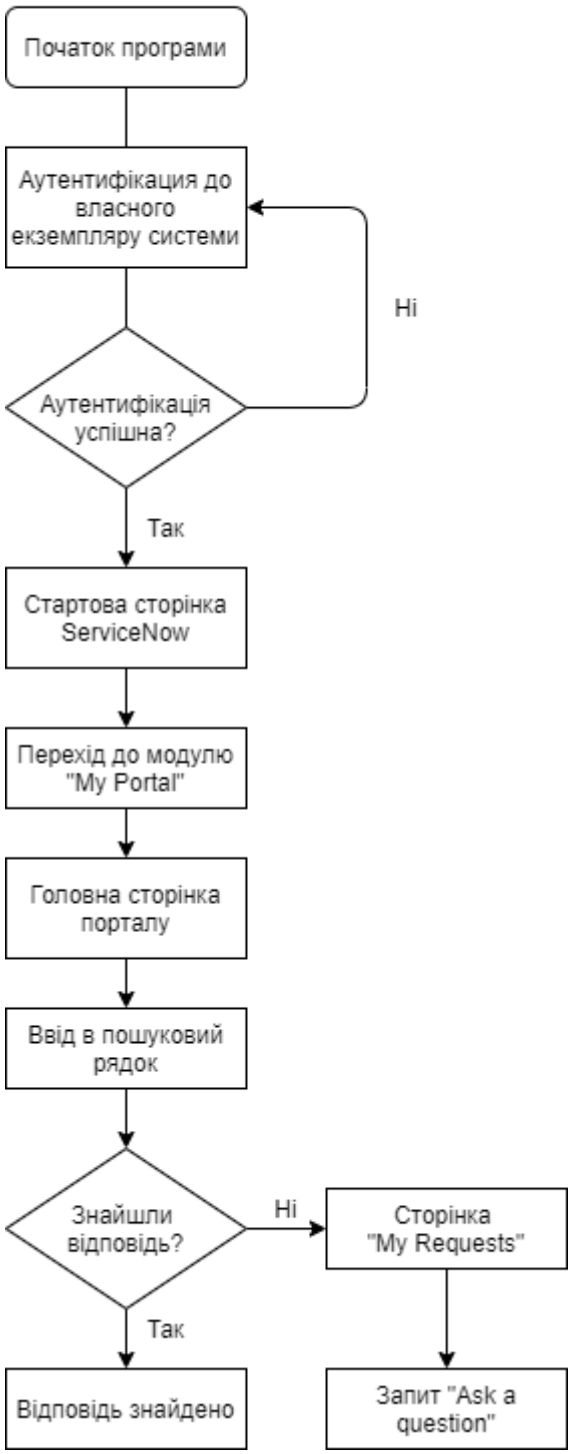
15. Простой туториал React Router v4 [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/post/329996/.

16. Управление инцидентами (Incident management) по ITIL [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://itsm365.ru/blog/articles/upravlenie-incidentami-incident-management-po-itil/.

17. Что такое Self-service системы? [Електронний ресурс] – Режим доступу до ресурсу: https://www.helpdeski.ru/tags/self-service_sistemy/.

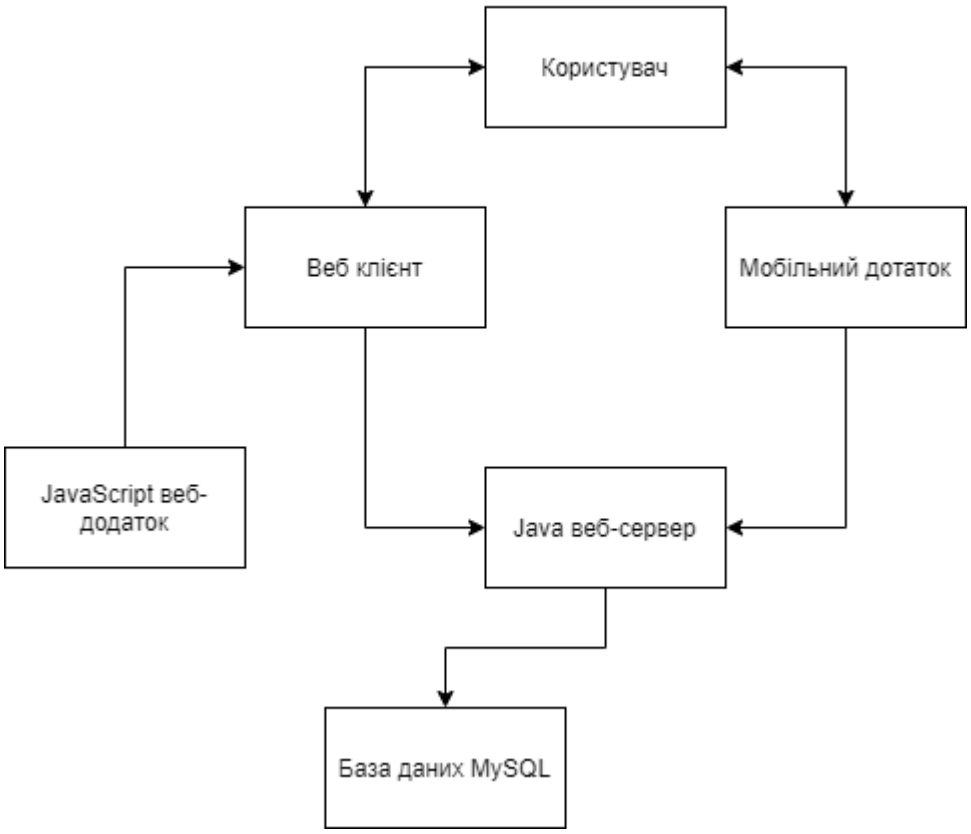
18. Что такое библиотека ITIL и зачем она нужна вашей компании [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/447276/.

Схема Алгоритму



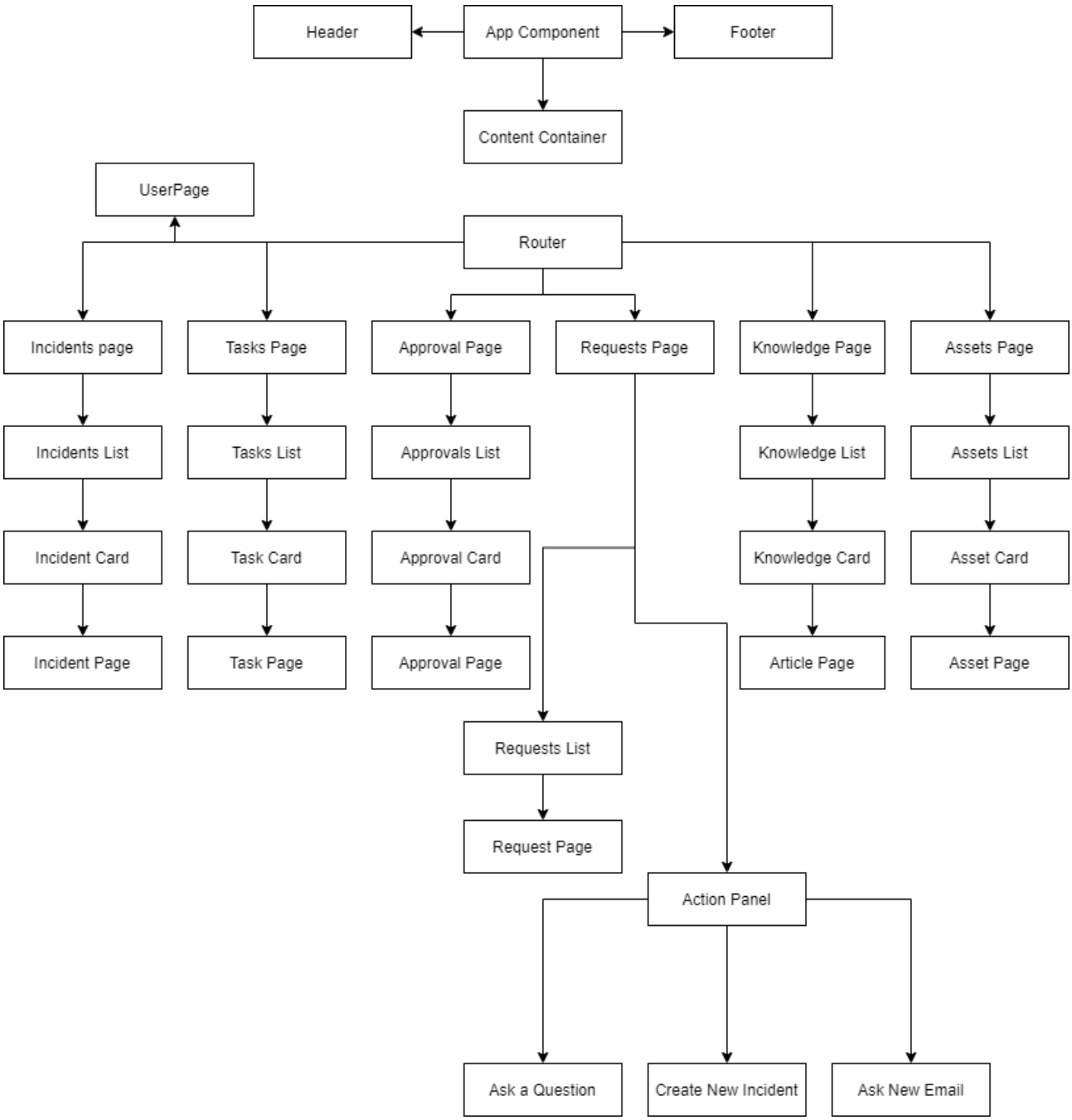
					ДП.6423.04.000 Д1		
Зм.		№ документа	Підп.	Дата			
					Автоматизована система менеджменту бізнес процесів на базі ITSM платформи Додаток А	Літ.	Аркуш 55 / 1
Н.конт		Симоненко В.П.				Т	НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 64

Структурна схема



					ДП.6423.05.000 Д2		
Зм.		№ документа	Підп.	Дата			
					Автоматизована система менеджменту бізнес процесів на базі ITSM платформи Додаток Б	Літ.	Аркуш 1 / 1
Н.конт		Симоненко В.П.				НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 64	

Діаграма компонентів



					ДП.6423.06.000 ДЗ		
Зм.		*№ документа*	*Підп.*	*Дата*			
					Автоматизована система менеджменту бізнес процесів на базі ITSM платформи	*Літ.*	*Аркуш*	
						Т	1	1
Н.конт		*Симоненко В.П.*			*Додаток В*	*НТУУ «КПІ імені Ігоря Сікорського», ФІОТ*		
						Група ІО - 64		